LAB 1

CONSTRUCT A SIMPLE NETWORK

Name: Trương Đặng Trúc Lâm

ID: B2111933

Group: M04

Submission: an ID_NAME_Lab01.pdf file describes clearly how did you solve the problem

Exercise 0: change the directory to your home directory

Answer: $cd

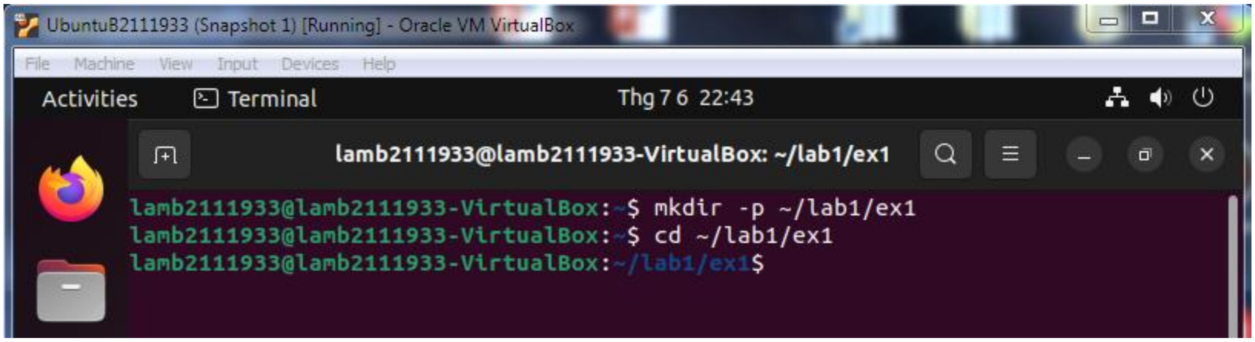
Exercise 1: Construct a simple network with two hosts connected to the same collision domain

Answer:

Create the directory **~/lab1/ex1**:

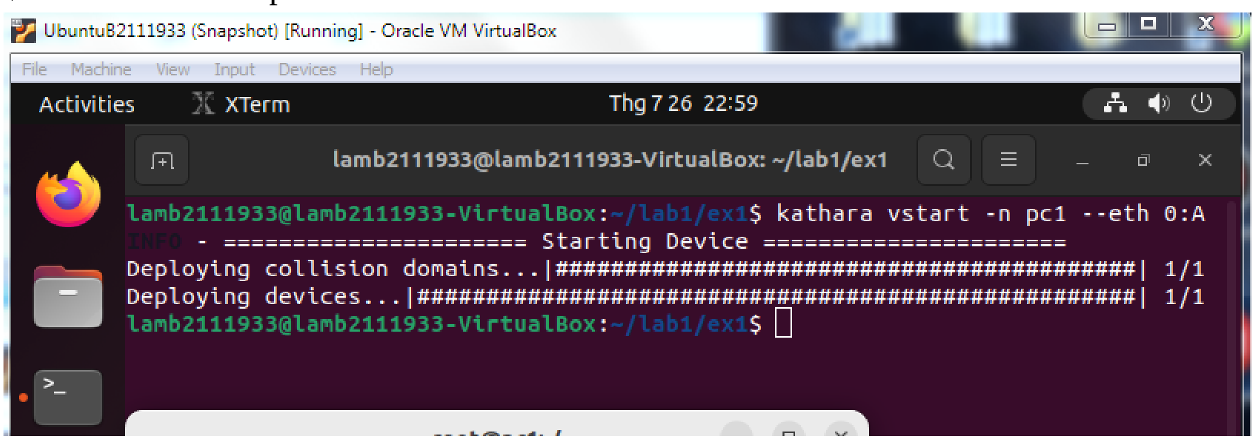
$ mkdir -p ~/lab1/ex1

$ cd ~/lab1/ex1



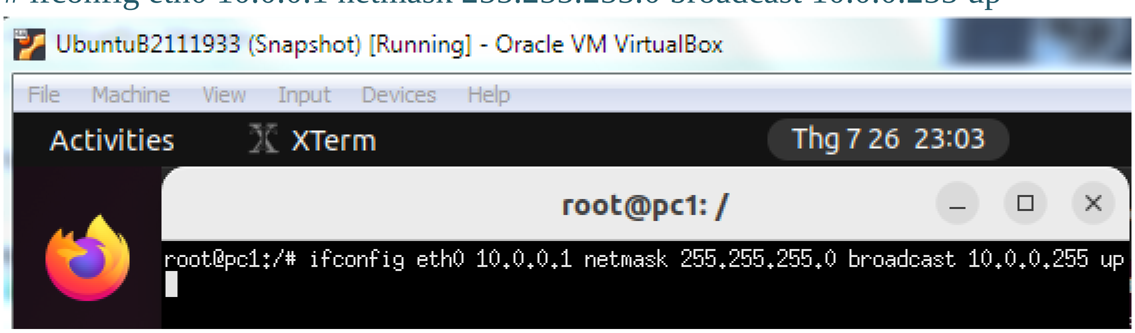
Start a new device called **pc1** and connected to the virtual collision domain **A**:

$ kathara vstart –n pc1 –eth 0:A



Configure network interfaces for **pc1**:

\# ifconfig eth0 10.0.0.1 netmask 255.255.255.0 broadcast 10.0.0.255 up

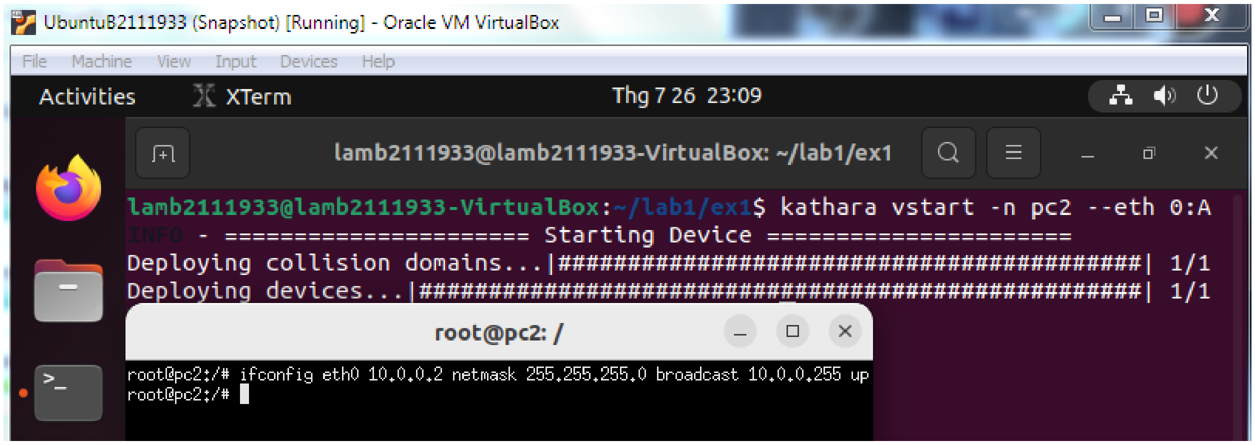


Start another device called **pc2** and connected to the virtual collision domain **A**

$ kathara vstart –n pc2 –eth 0:A

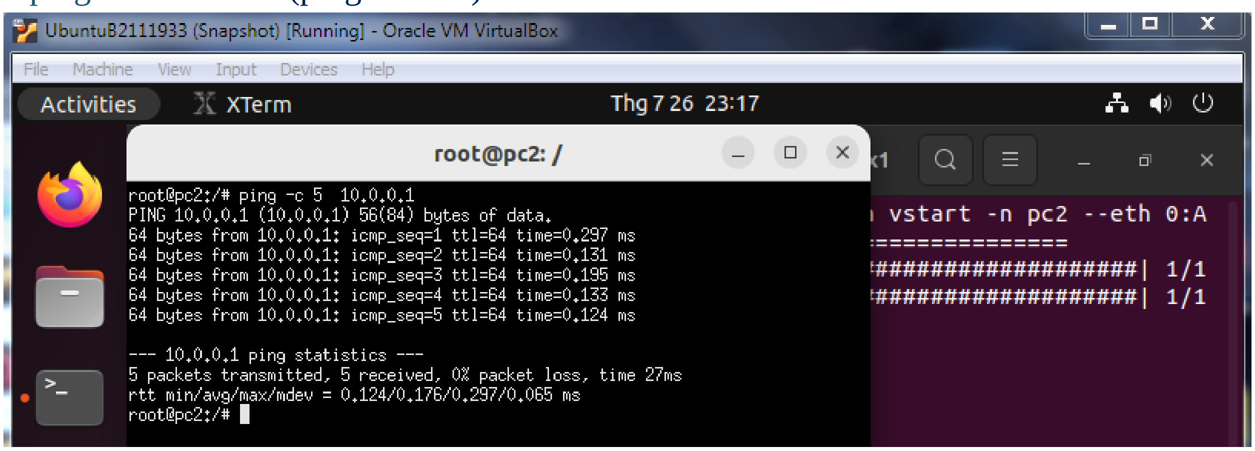
Then configure it's network interfaces:

ifconfig eth0 10.0.0.2 netmask 255.255.255.0 broadcast 10.0.0.255 up

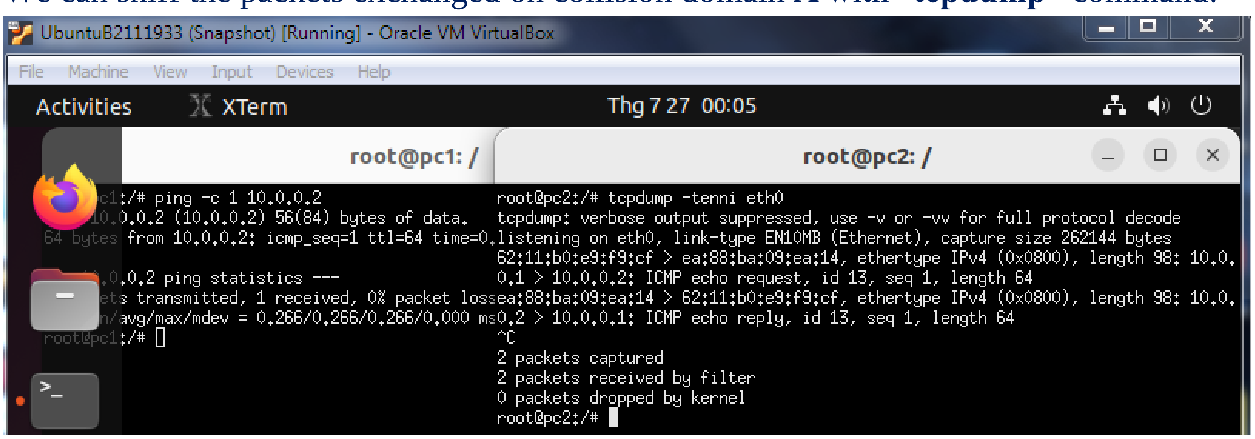


Test connectivity from **pc2** to **pc1**:

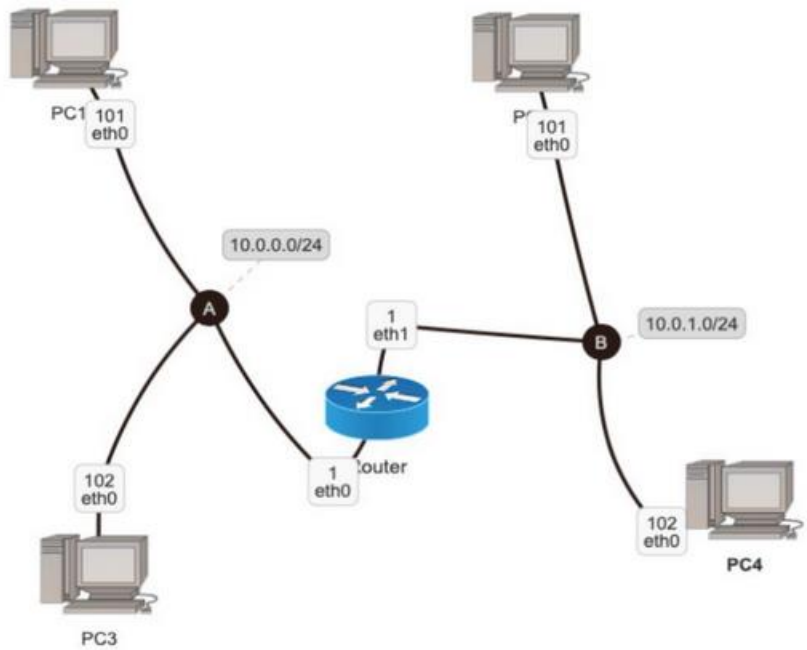
ping –c 5 10.0.0.1 (ping 5 times)



We can sniff the packets exchanged on collision domain **A** with "**tcpdump**" command:



Exercise 2: Construct the network



Answer:

Prepare a lab for **Exersise 2**:

$ mkdir pc1 pc2 pc3 pc4 router1 shared

$ gedit lab.conf

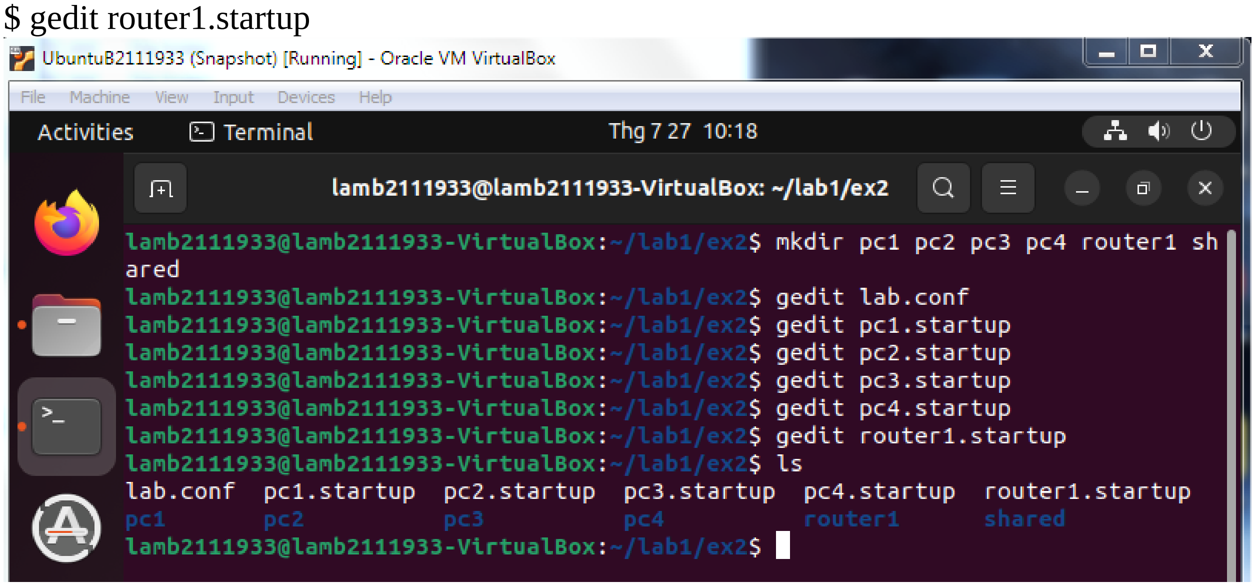
$ gedit pc1.startup

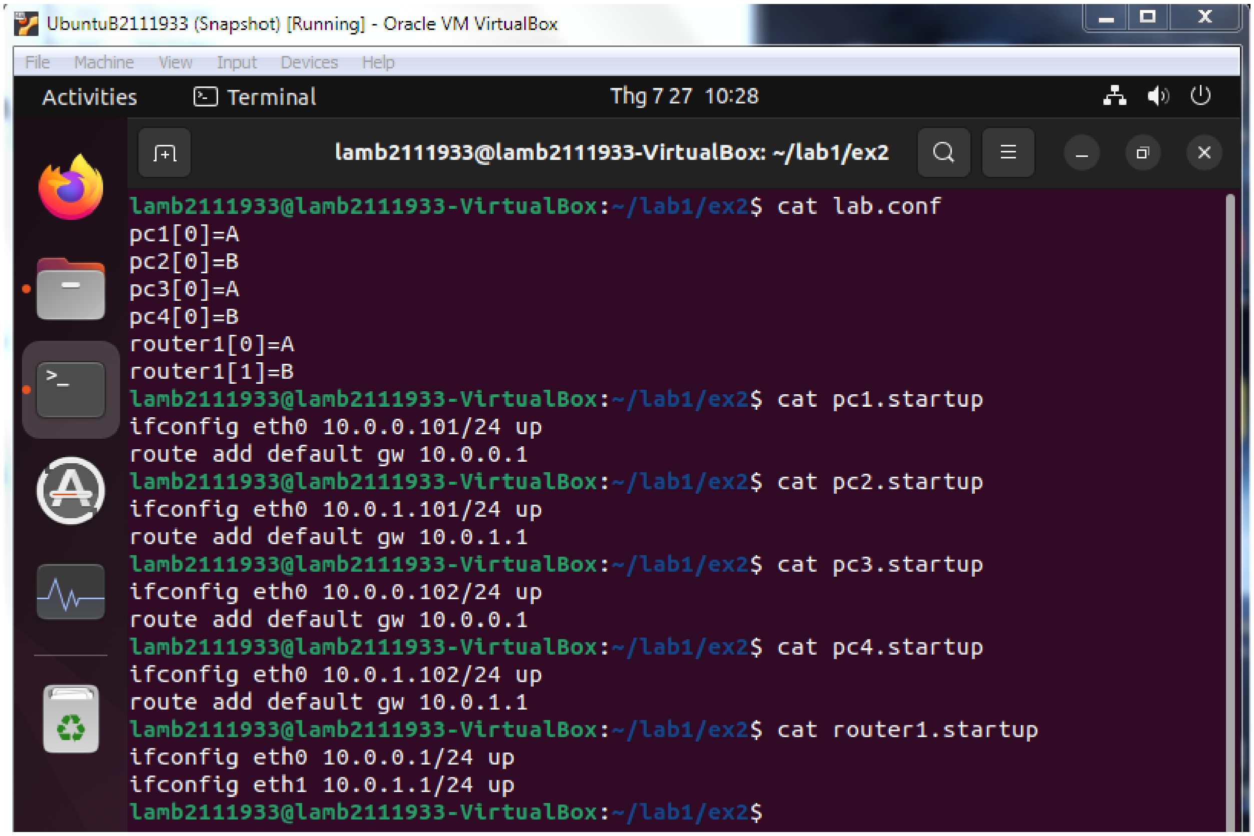
$ gedit pc2.startup

$ gedit pc3.startup

$ gedit pc4.startup

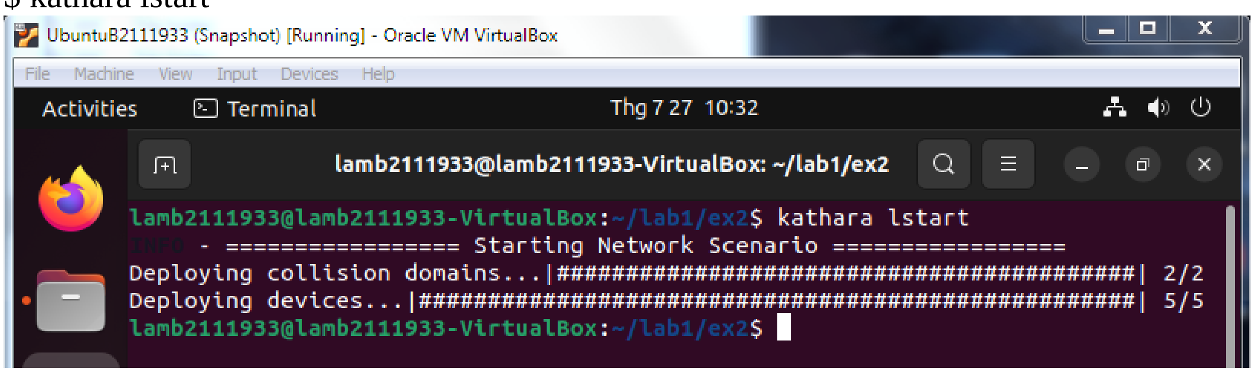
$ gedit router1.startup



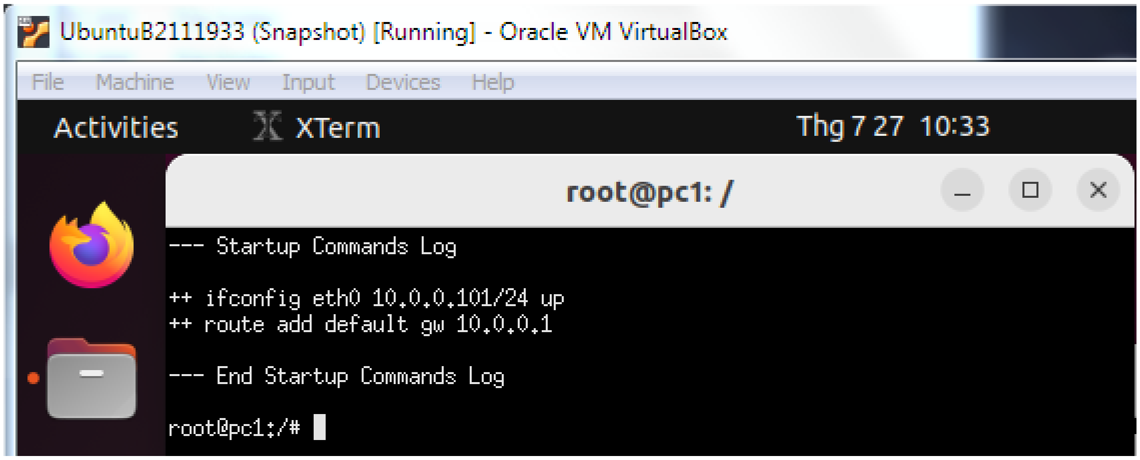


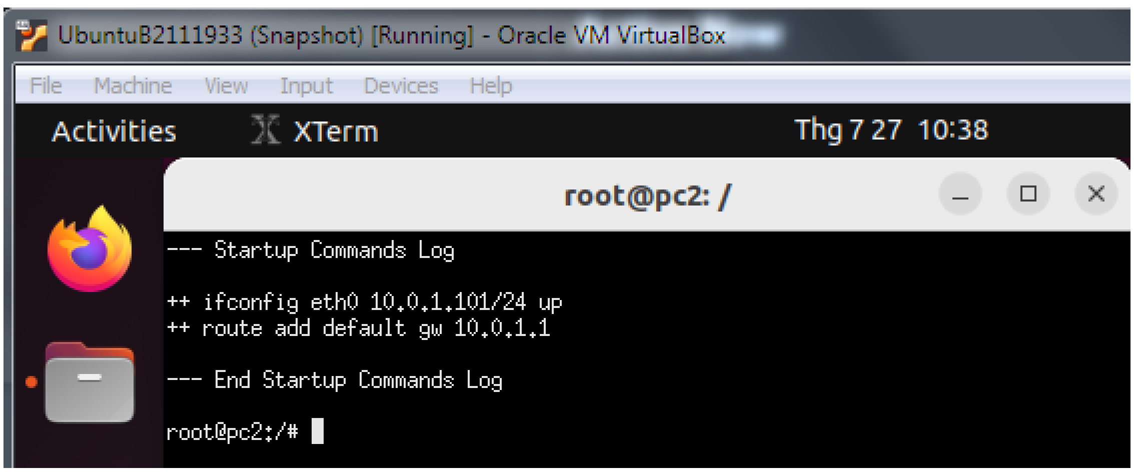
Start the prepared lab:

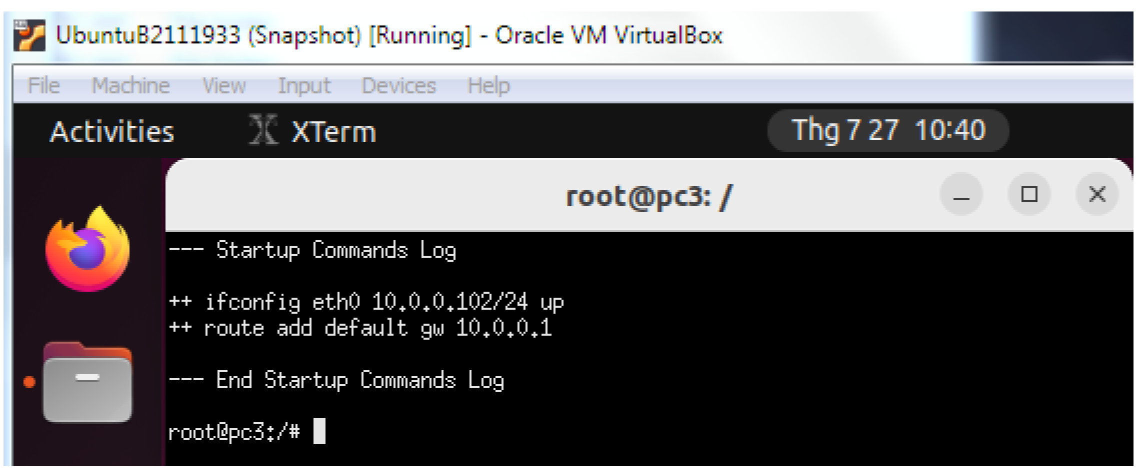
$ kathara lstart

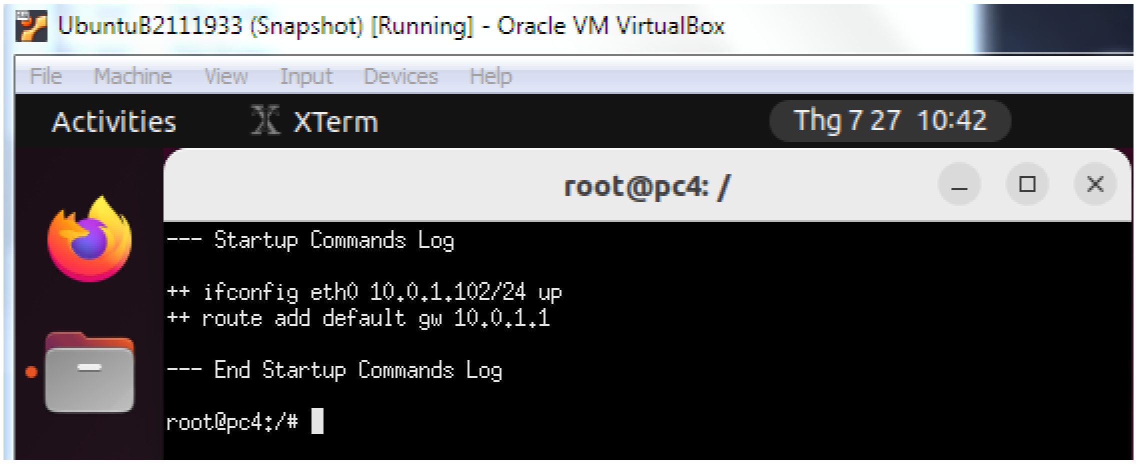


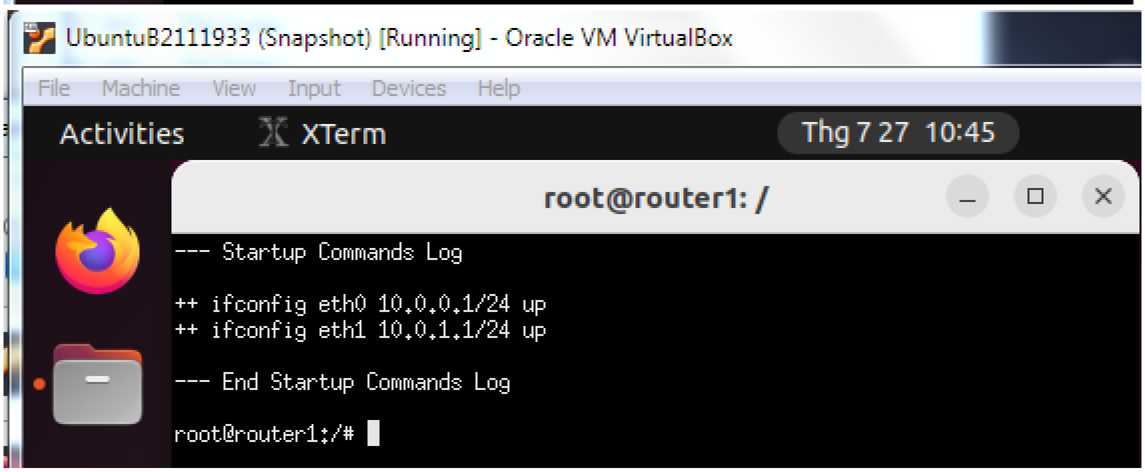
It will startup commands log:



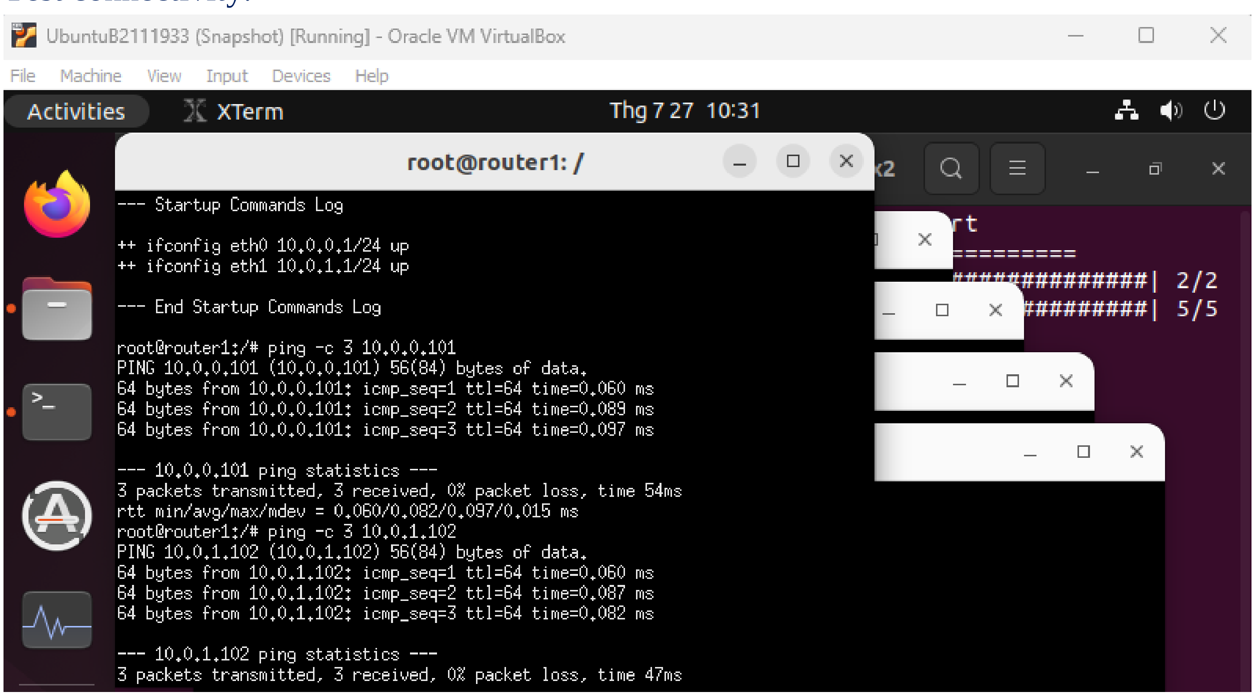


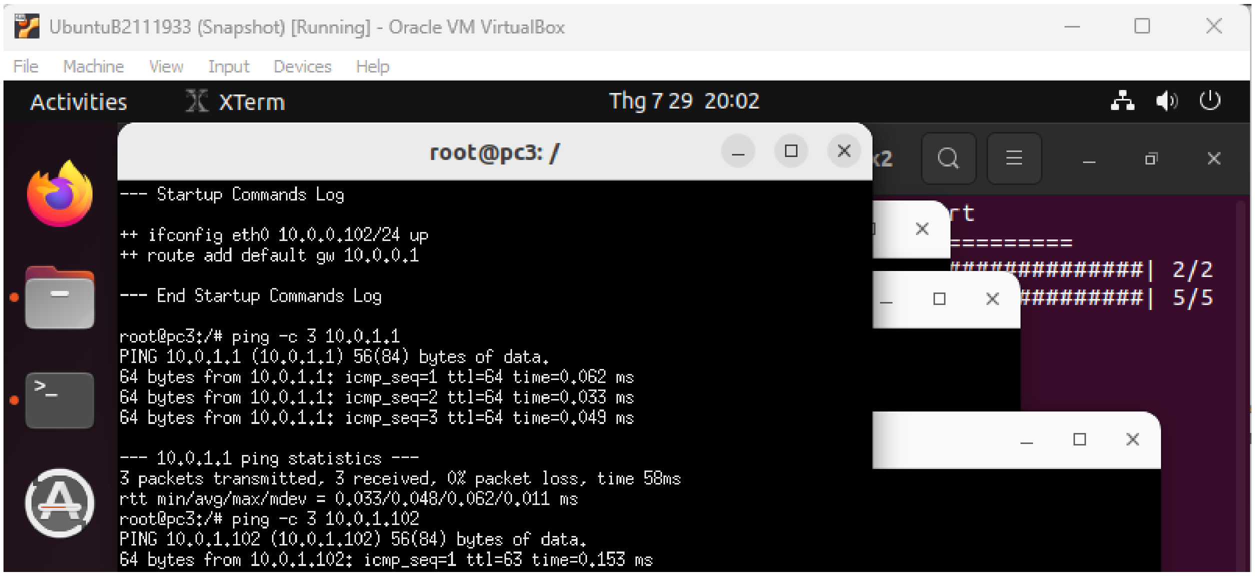




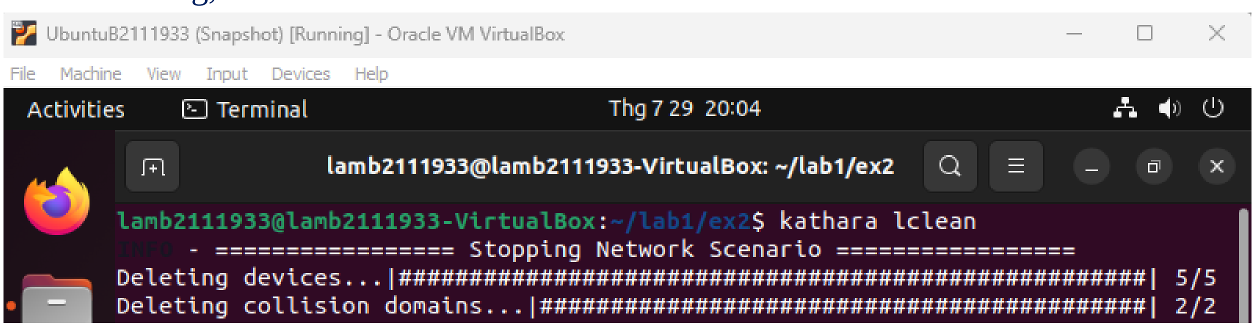


Test connectivity:

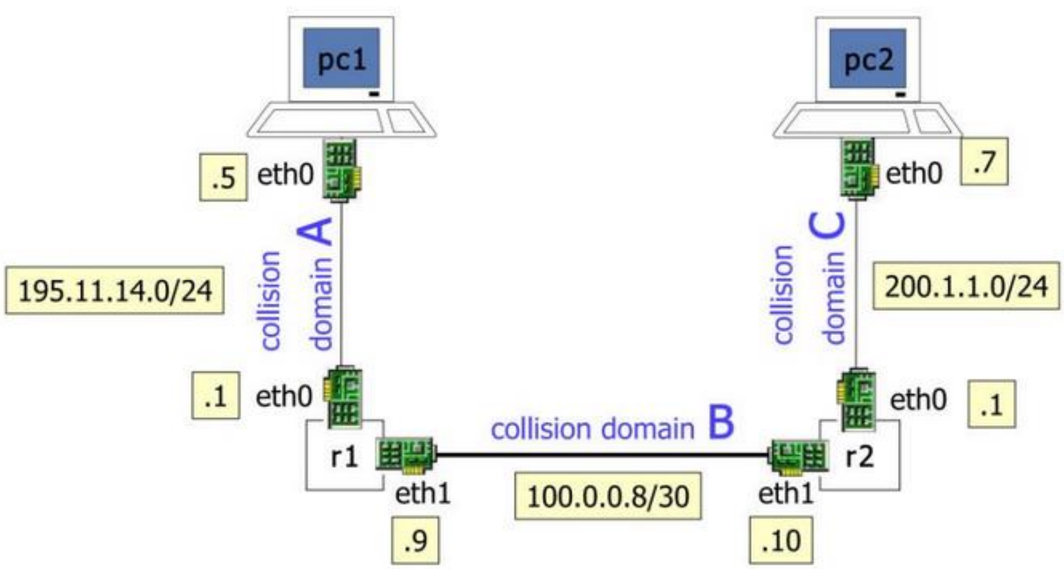




Before leaving, remember to halts all the devices of this lab:



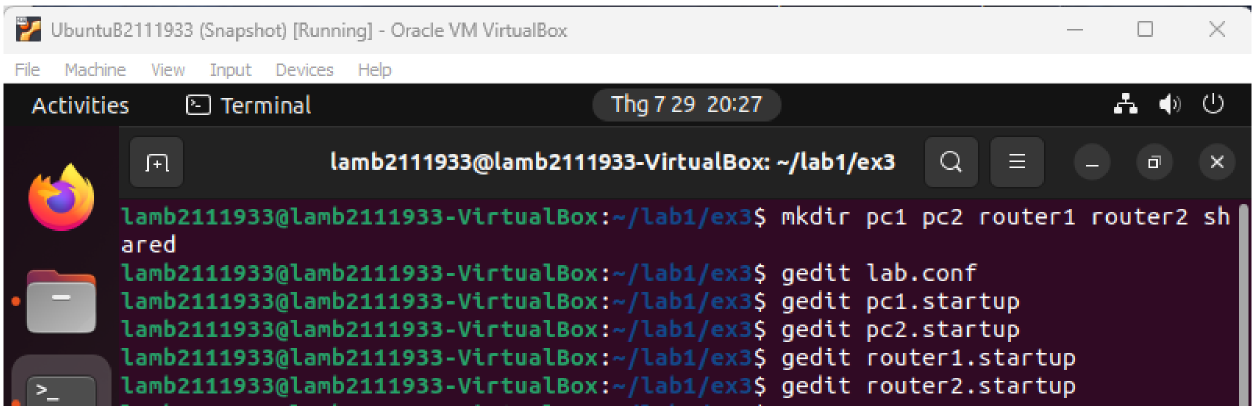
Exercise 3: Construct the network

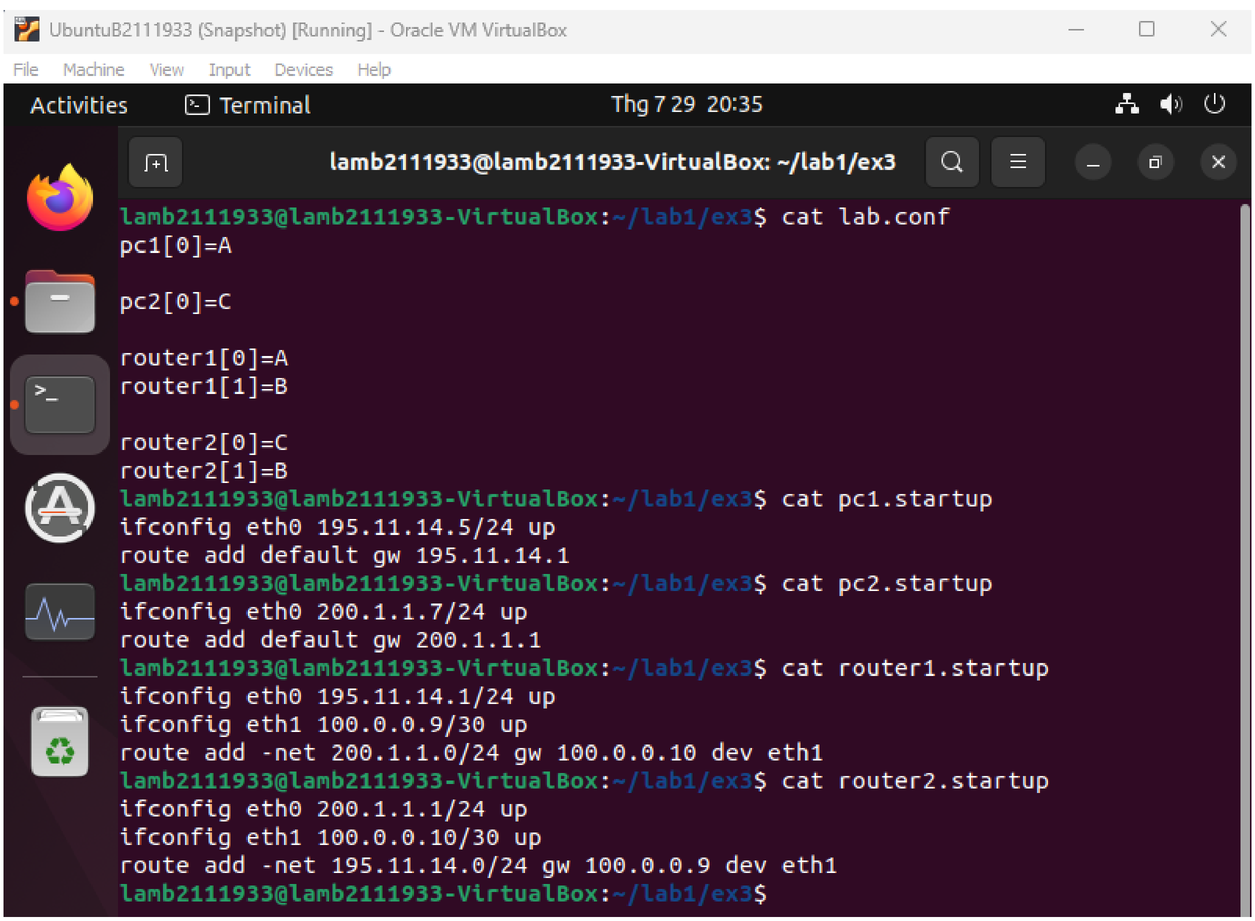


Answer:

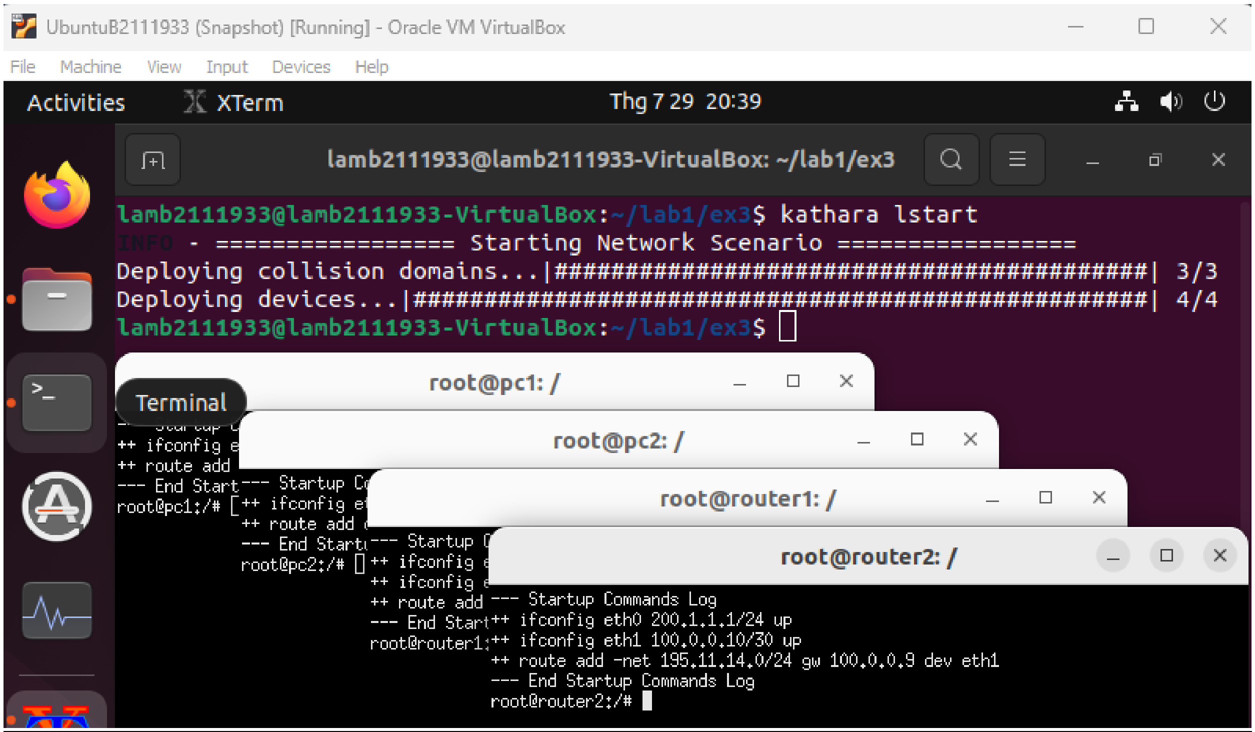
Prepare a lab for **Exersise 3**:

$ mkdir pc1 pc2 router1 router2 shared
$ gedit lab.conf
$ gedit pc1.startup
$ gedit pc2.startup
$ gedit router1.startup
$ gedit router2.startup

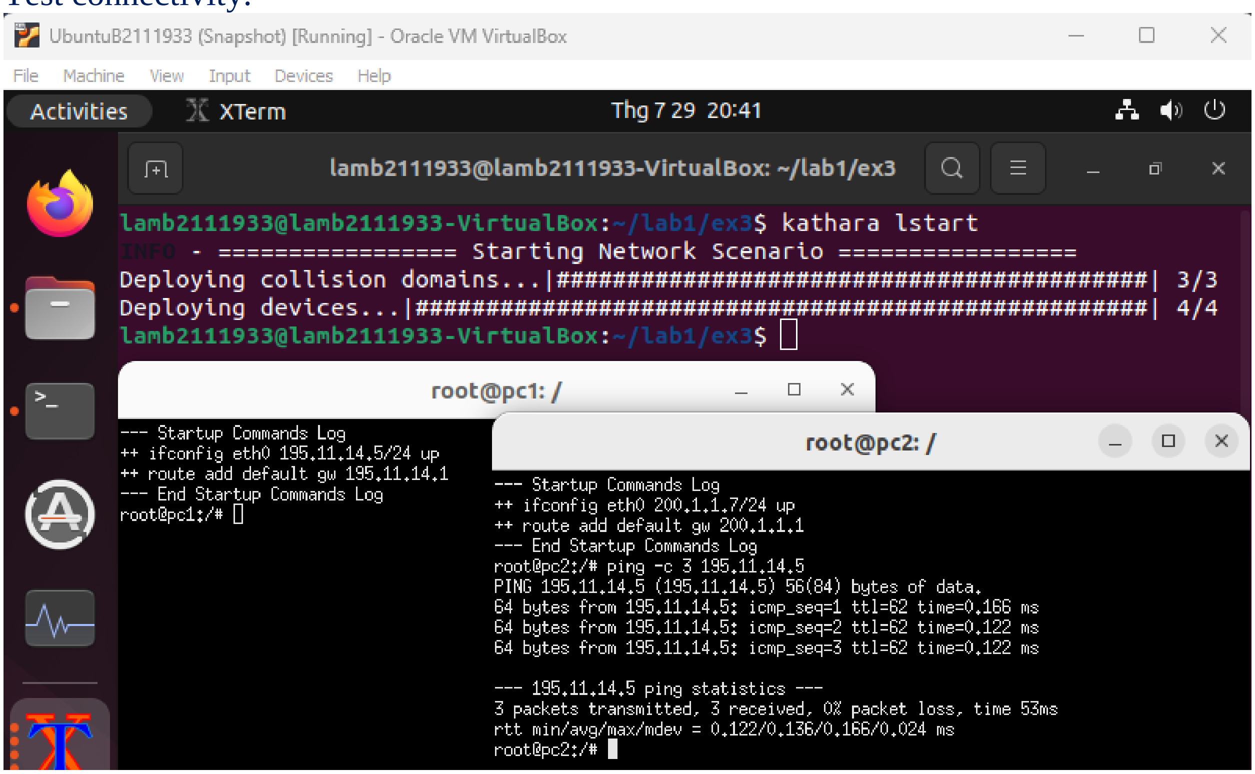




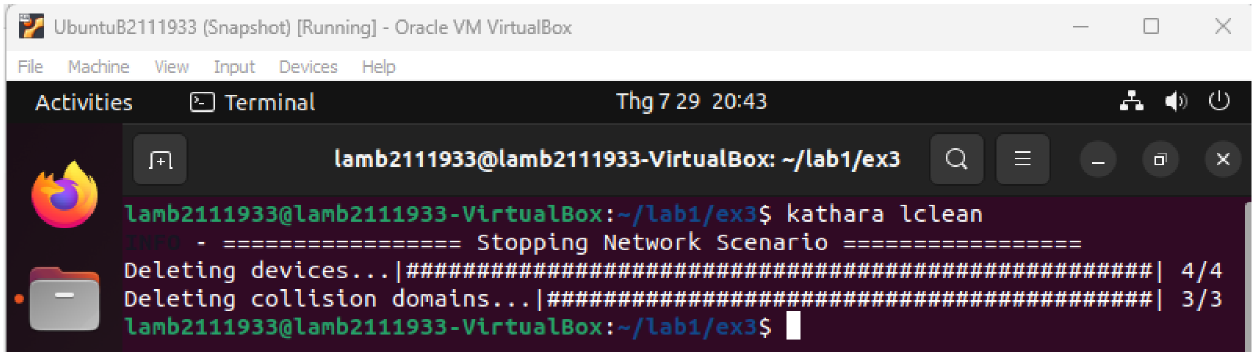
Start the prepared lab:



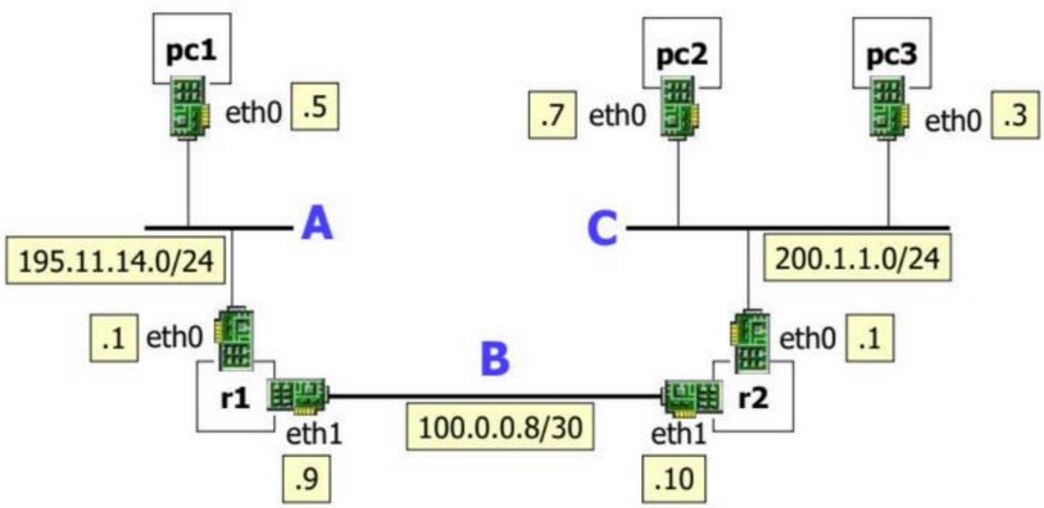
Test connectivity:



Before leaving, remember to halts all the devices of this lab:



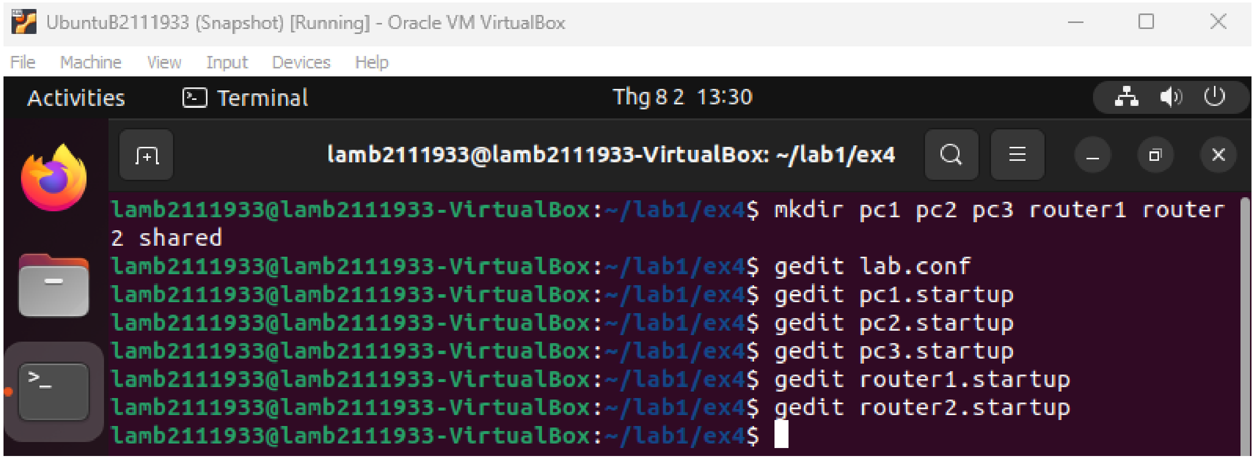
Exercise 4: study arp protocol

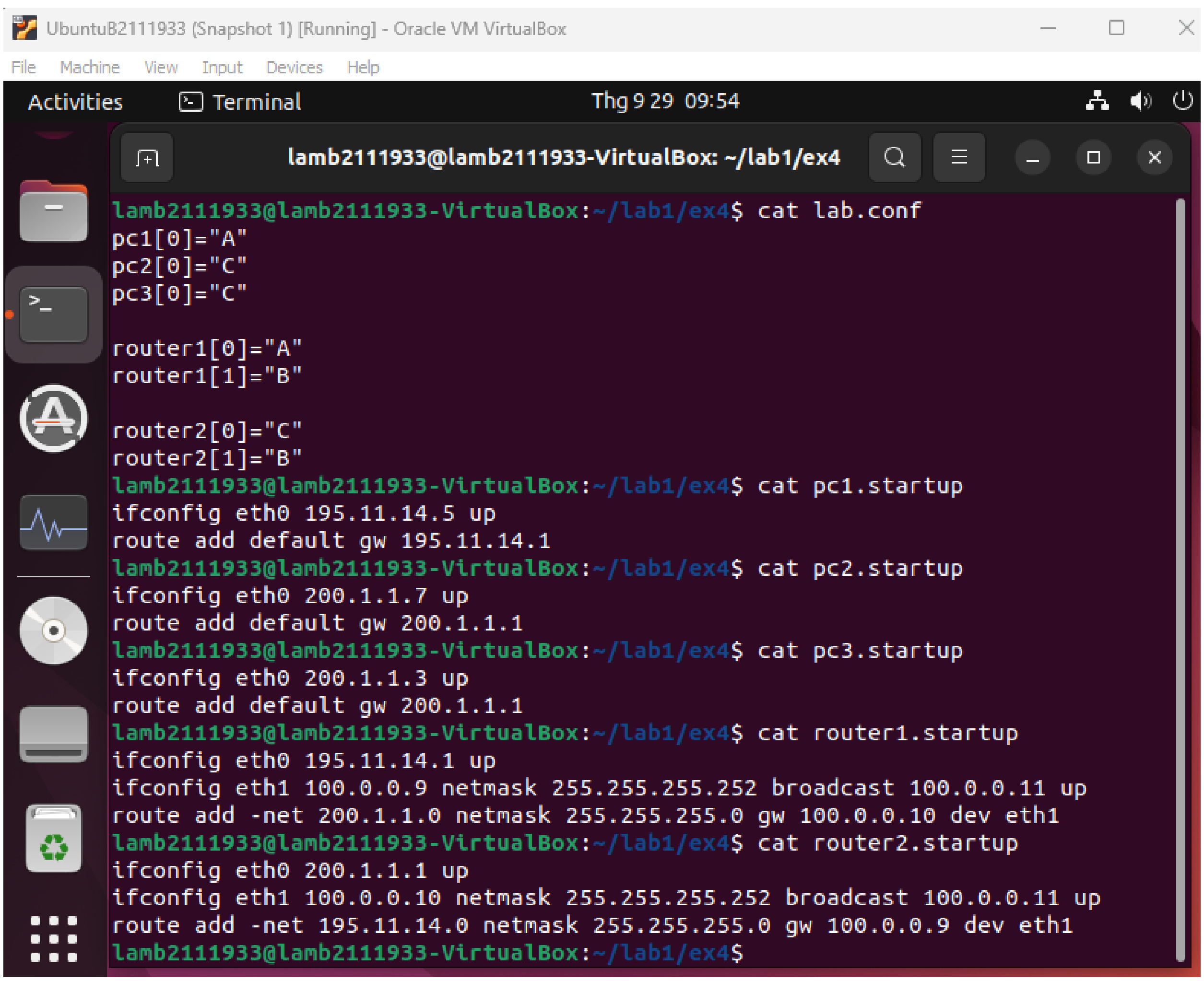


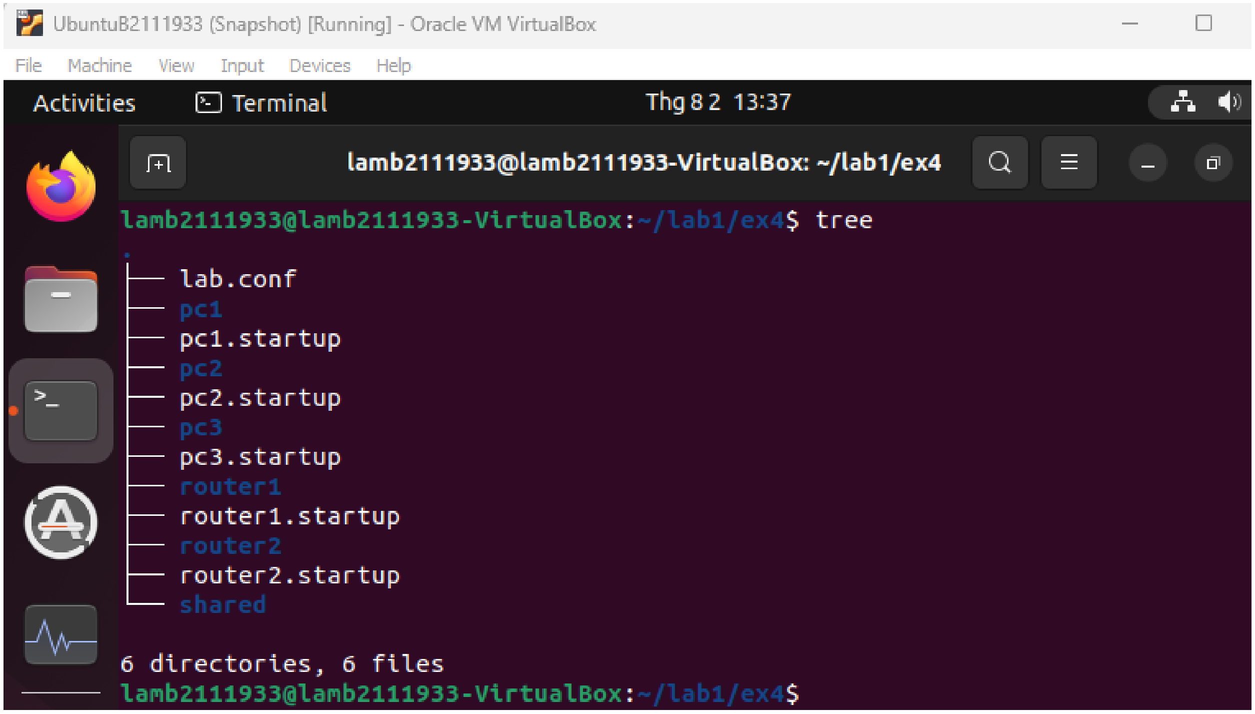
Answer:

Prepare a lab for **Exersise 4**:

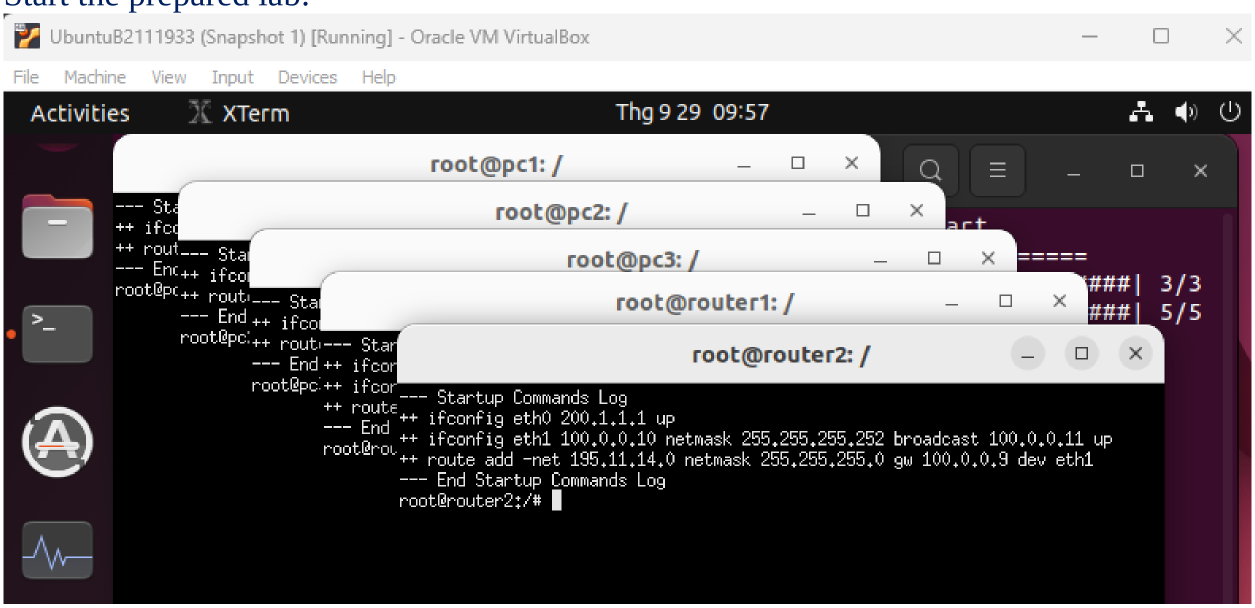
$ mkdir pc1 pc2 pc3 router1 router2 shared
$ gedit lab.conf
$ gedit pc1.startup
$ gedit pc2.startup
$ gedit pc3.startup
$ gedit router1.startup
$ gedit router2.startup


UbuntuB2111933 (Snapshot) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Thg 8 2 13:30
lamb2111933@lamb2111933-VirtualBox: ~/lab1/ex4
lamb2111933@lamb2111933-VirtualBox:~/lab1/ex4$ mkdir pc1 pc2 pc3 router1 router
2 shared
lamb2111933@lamb2111933-VirtualBox:~/lab1/ex4$ gedit lab.conf
lamb2111933@lamb2111933-VirtualBox:~/lab1/ex4$ gedit pc1.startup
lamb2111933@lamb2111933-VirtualBox:~/lab1/ex4$ gedit pc2.startup
lamb2111933@lamb2111933-VirtualBox:~/lab1/ex4$ gedit pc3.startup
lamb2111933@lamb2111933-VirtualBox:~/lab1/ex4$ gedit router1.startup
lamb2111933@lamb2111933-VirtualBox:~/lab1/ex4$ gedit router2.startup
lamb2111933@lamb2111933-VirtualBox:~/lab1/ex4$


UbuntuB2111933 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Thg 9 29 09:54
lamb2111933@lamb2111933-VirtualBox: ~/lab1/ex4
lamb2111933@lamb2111933-VirtualBox:~/lab1/ex4$ cat lab.conf
pc1[0]="A"
pc2[0]="C"
pc3[0]="C"
router1[0]="A"
router1[1]="B"
router2[0]="C"
router2[1]="B"
lamb2111933@lamb2111933-VirtualBox:~/lab1/ex4$ cat pc1.startup
ifconfig eth0 195.11.14.5 up
route add default gw 195.11.14.1
lamb2111933@lamb2111933-VirtualBox:~/lab1/ex4$ cat pc2.startup
ifconfig eth0 200.1.1.7 up
route add default gw 200.1.1.1
lamb2111933@lamb2111933-VirtualBox:~/lab1/ex4$ cat pc3.startup
ifconfig eth0 200.1.1.3 up
route add default gw 200.1.1.1
lamb2111933@lamb2111933-VirtualBox:~/lab1/ex4$ cat router1.startup
ifconfig eth0 195.11.14.1 up
ifconfig eth1 100.0.0.9 netmask 255.255.255.252 broadcast 100.0.0.11 up
route add -net 200.1.1.0 netmask 255.255.255.0 gw 100.0.0.10 dev eth1
lamb2111933@lamb2111933-VirtualBox:~/lab1/ex4$ cat router2.startup
ifconfig eth0 200.1.1.1 up
ifconfig eth1 100.0.0.10 netmask 255.255.255.252 broadcast 100.0.0.11 up
route add -net 195.11.14.0 netmask 255.255.255.0 gw 100.0.0.9 dev eth1
lamb2111933@lamb2111933-VirtualBox:~/lab1/ex4$

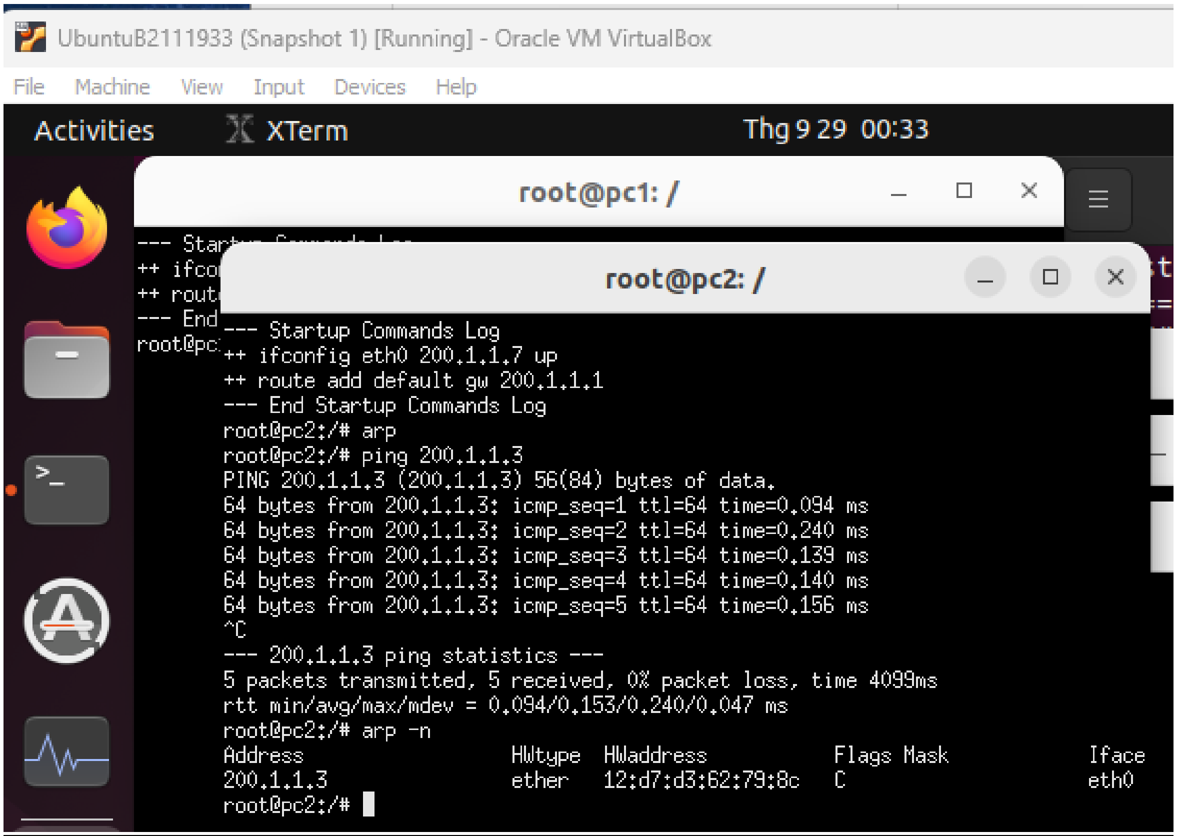


Start the prepared lab:

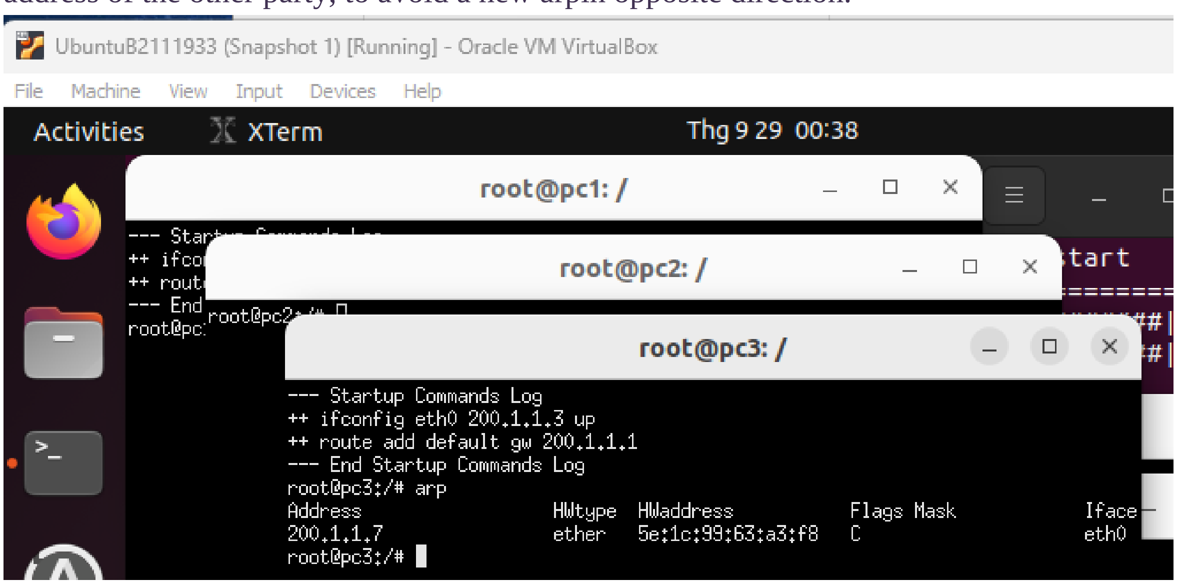


Inspect the arp cache (local traffic):

Use pc2 ping pc3 (local network) => Address resolution results are stored in the arp cache



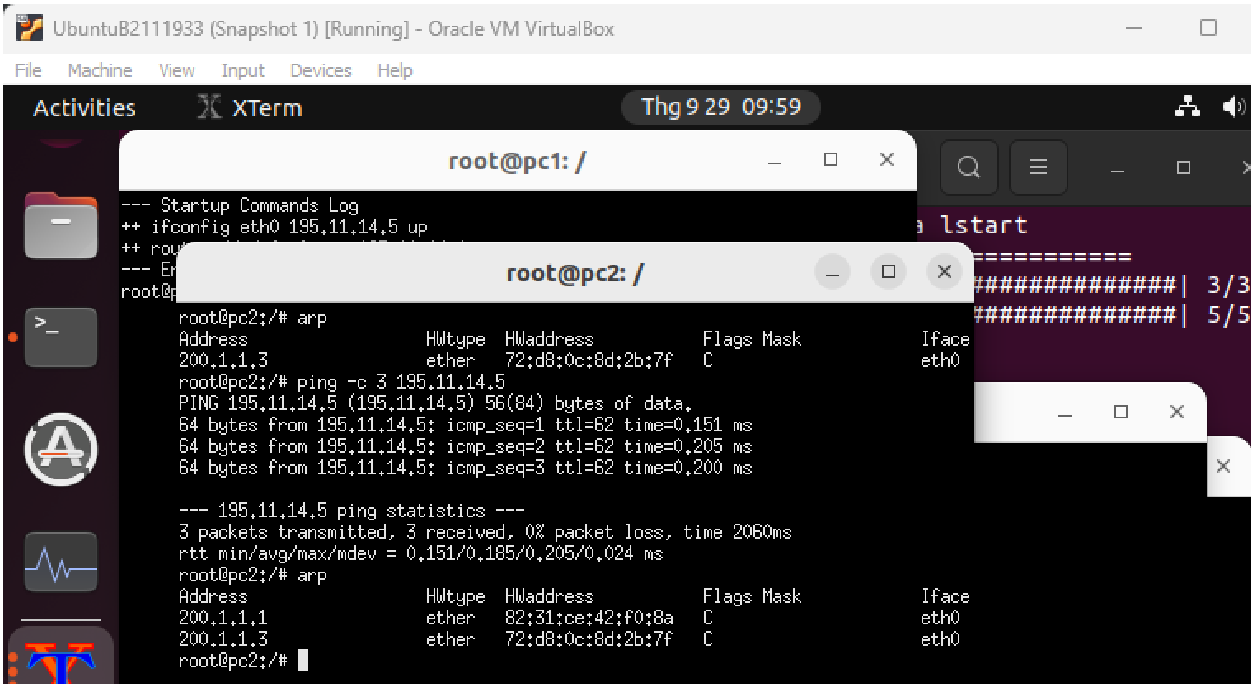
Communications are usually bi-directional. The receiver of the arp request learns the mac address of the other party, to avoid a new arpin opposite direction.



Inspect the arp cache (non local traffic):

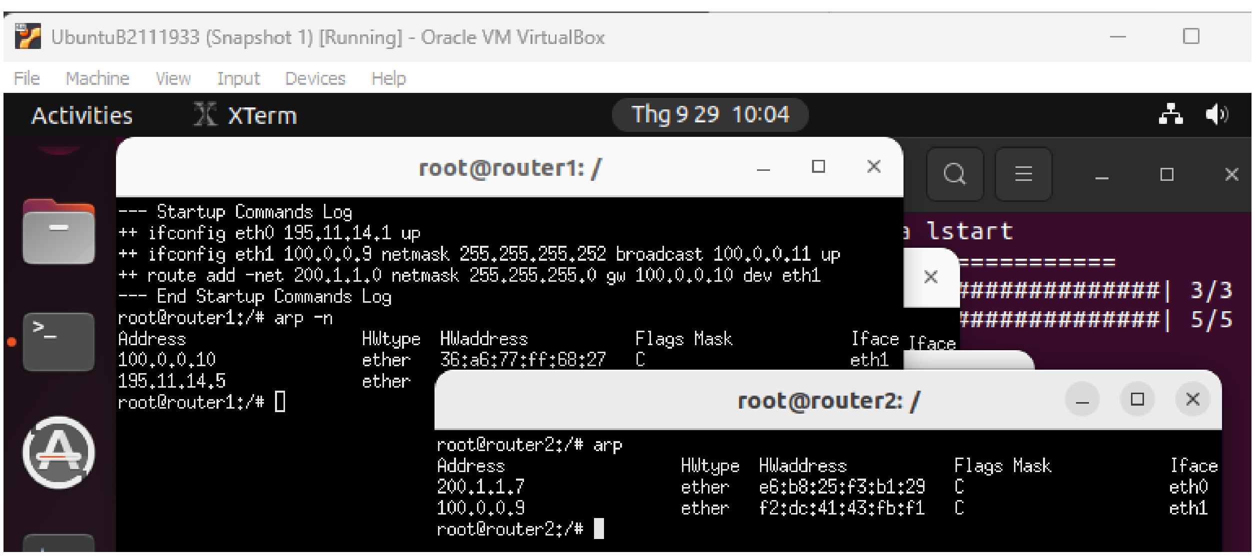
Use pc2 ping pc1 (non local network)

=> When ip traffic is addressed outside the local network, the sender needs the mac address of the router, arp requests can get replies only within the local network.



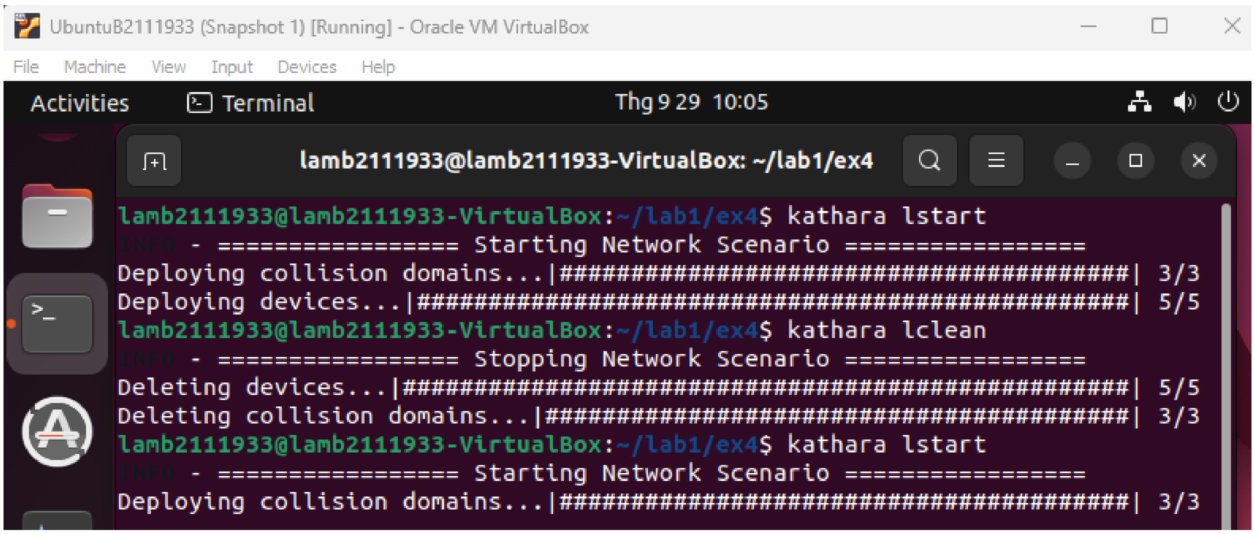
Inspect arp of the router:

- Routers perform arp too (hence have arp caches) anytime they have to send ip packets on an ethernet LAN.

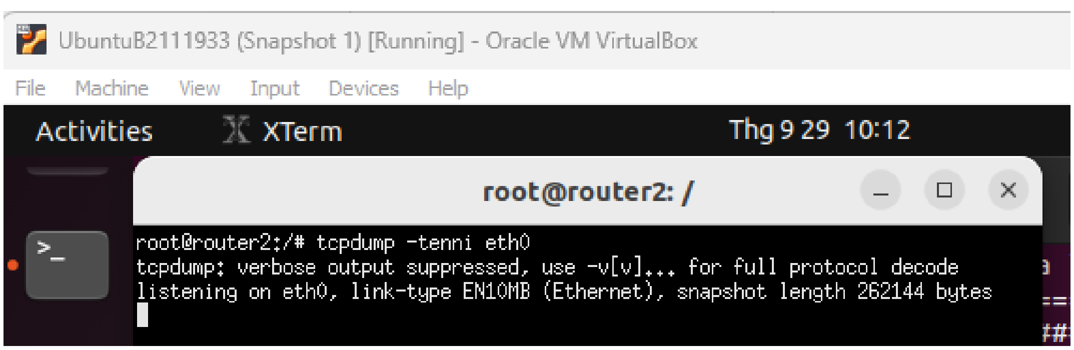


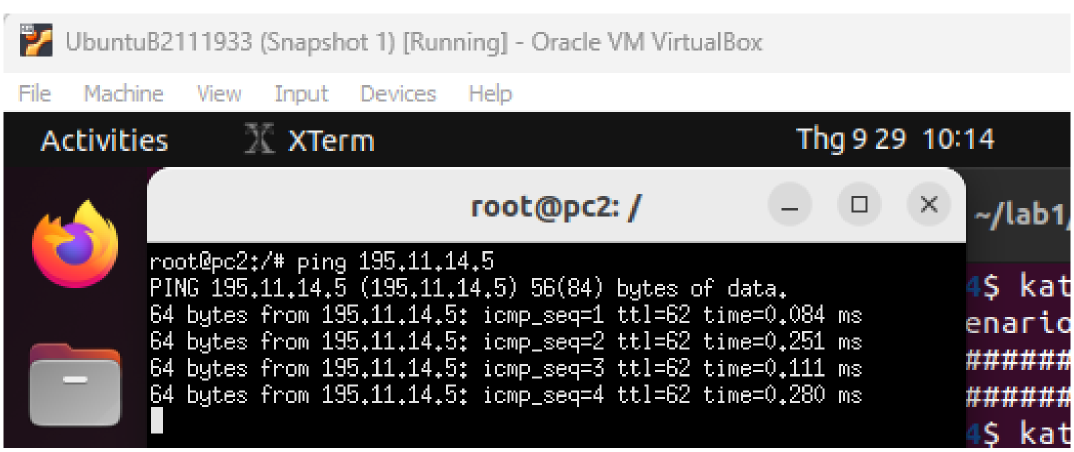
Sniff arp traffic:

Restart the lab in order to clear arp caches

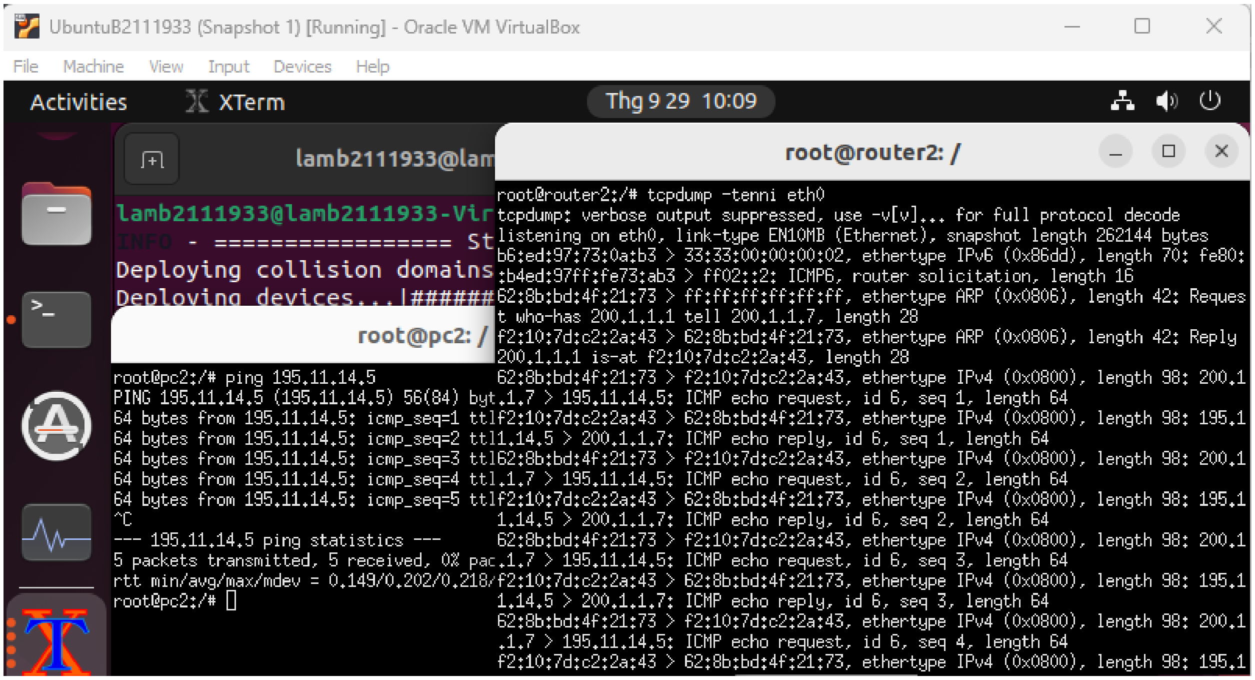


Start to sniff:

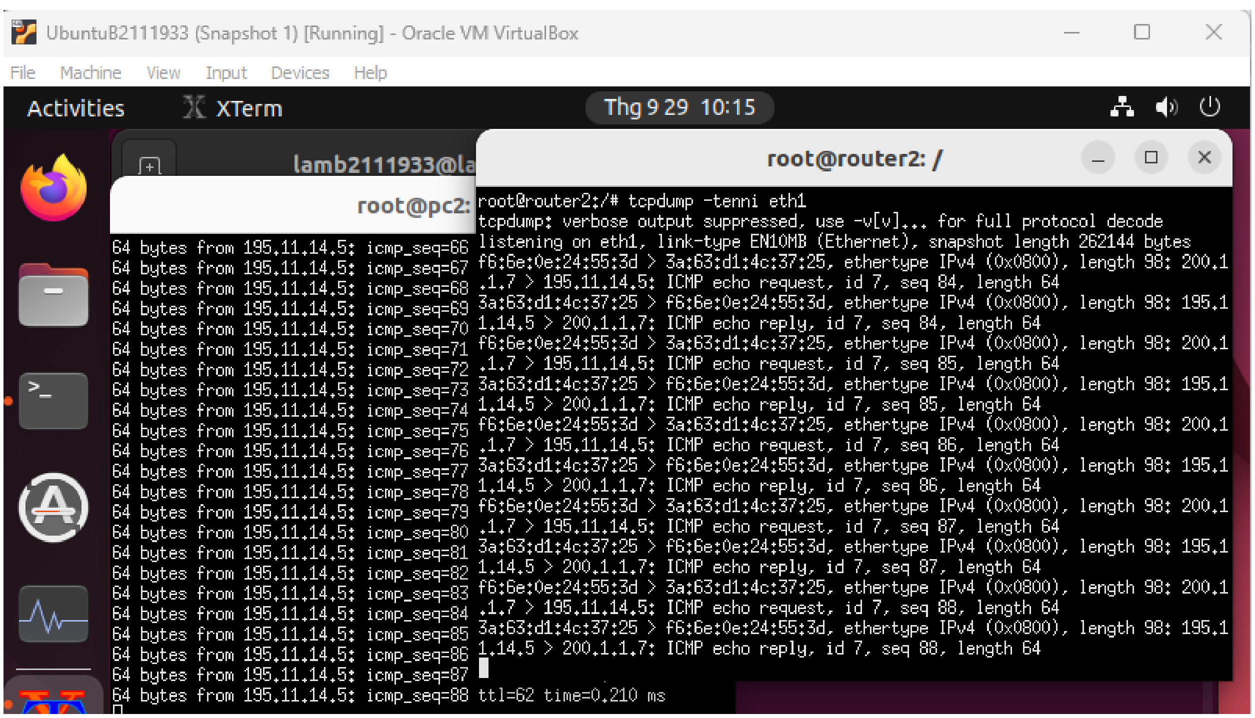




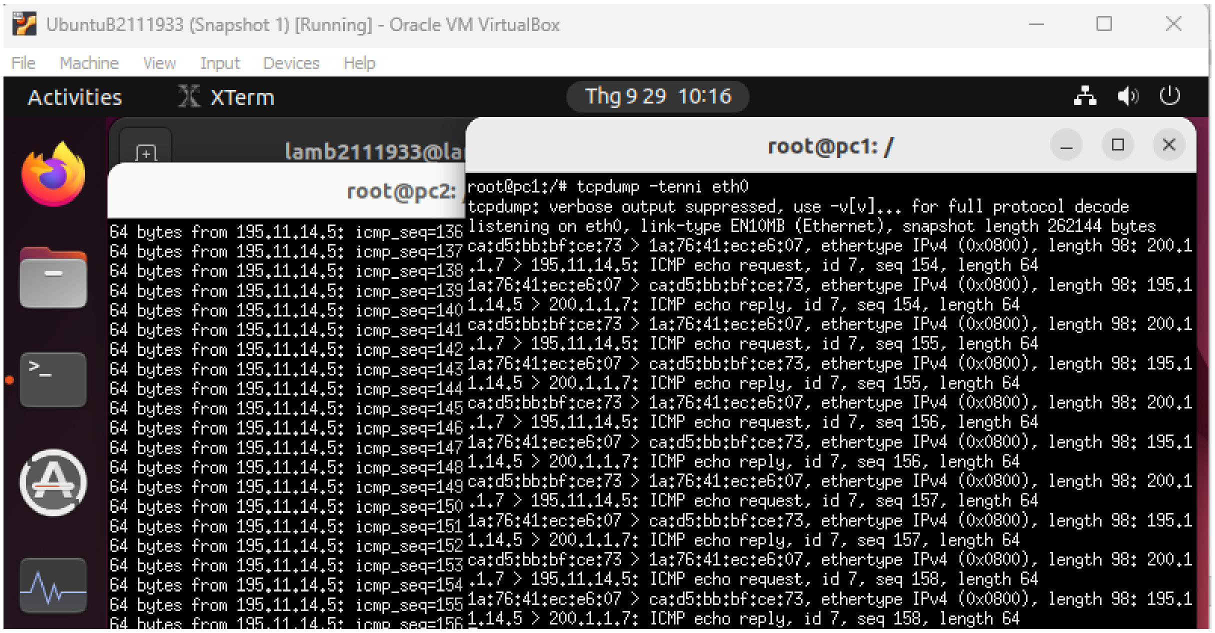
On collision domain C:



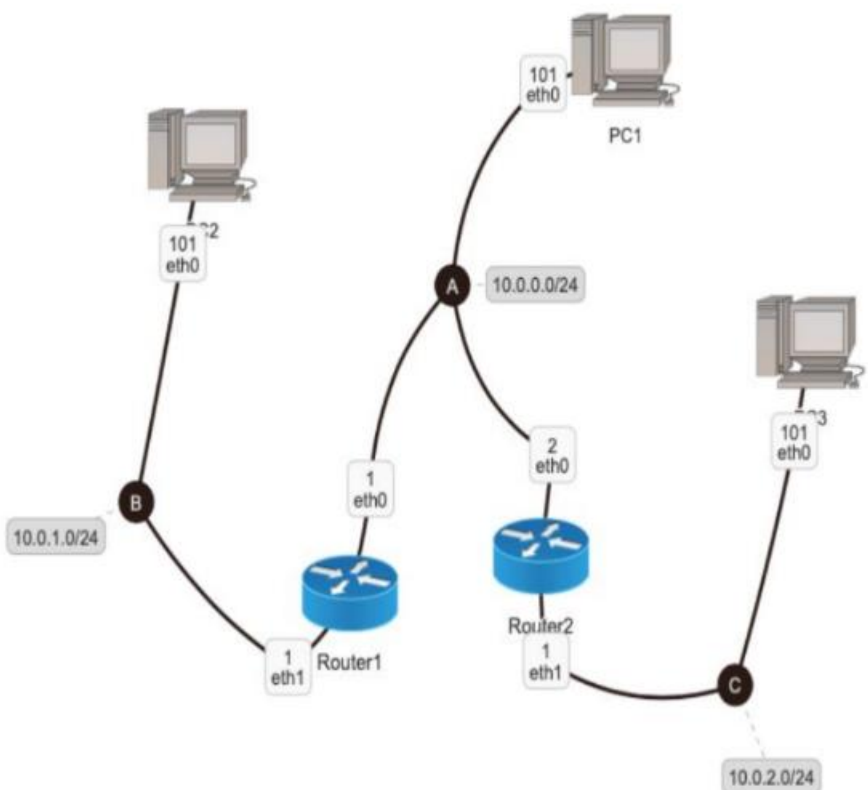
On collision domain B:



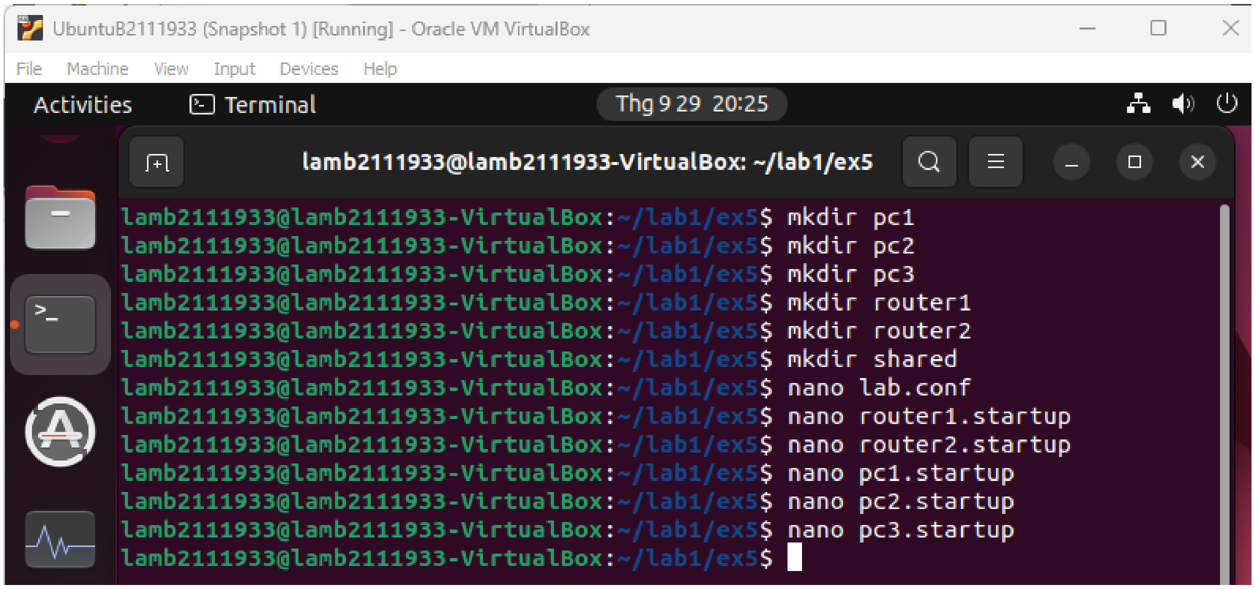
On collision domain A:



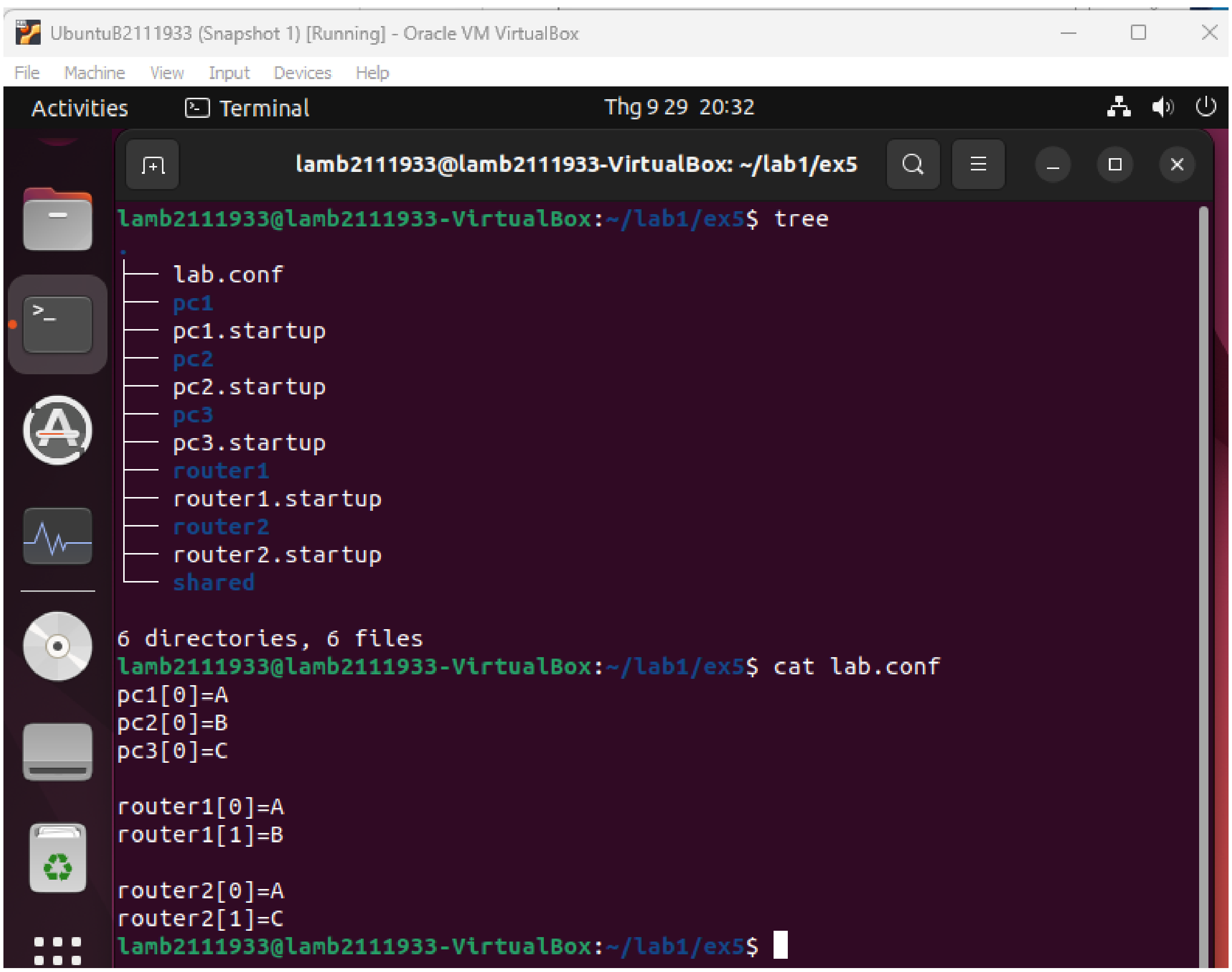
==**Exercise 5:**== Construct the network following the picture

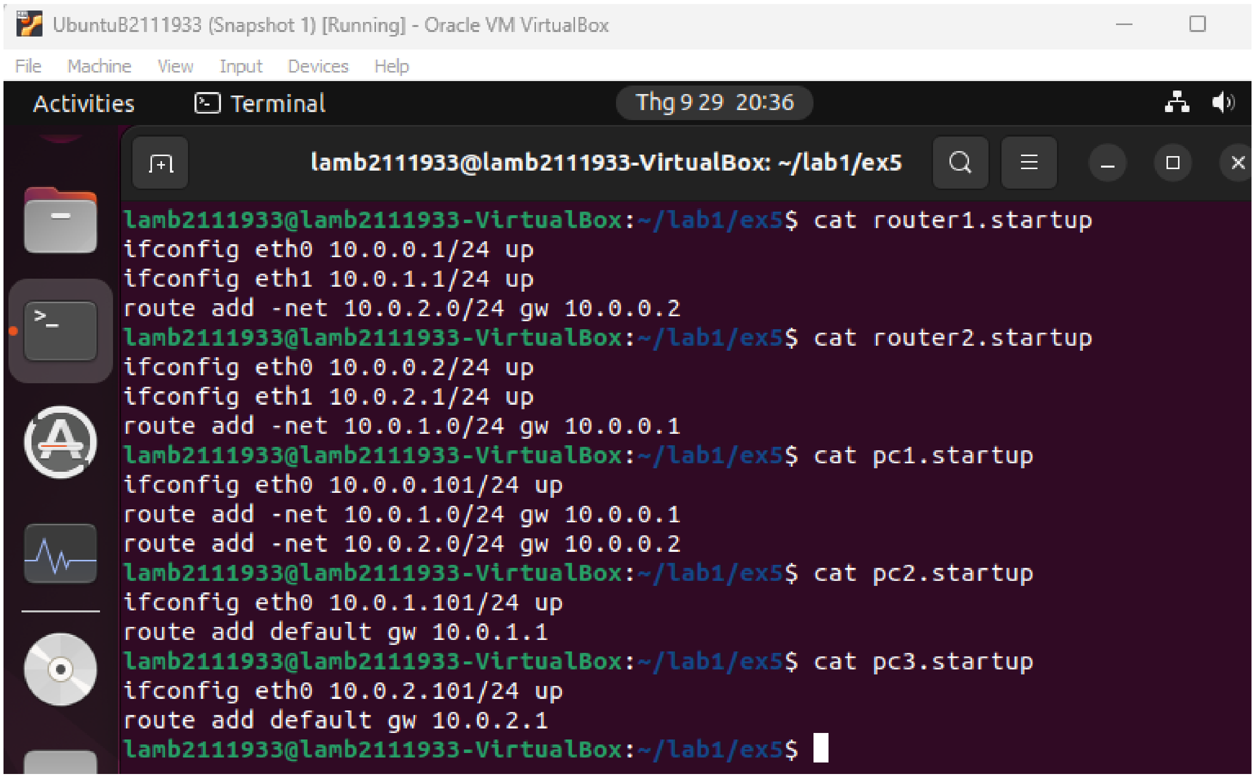


Create folders and files for exercise 5:

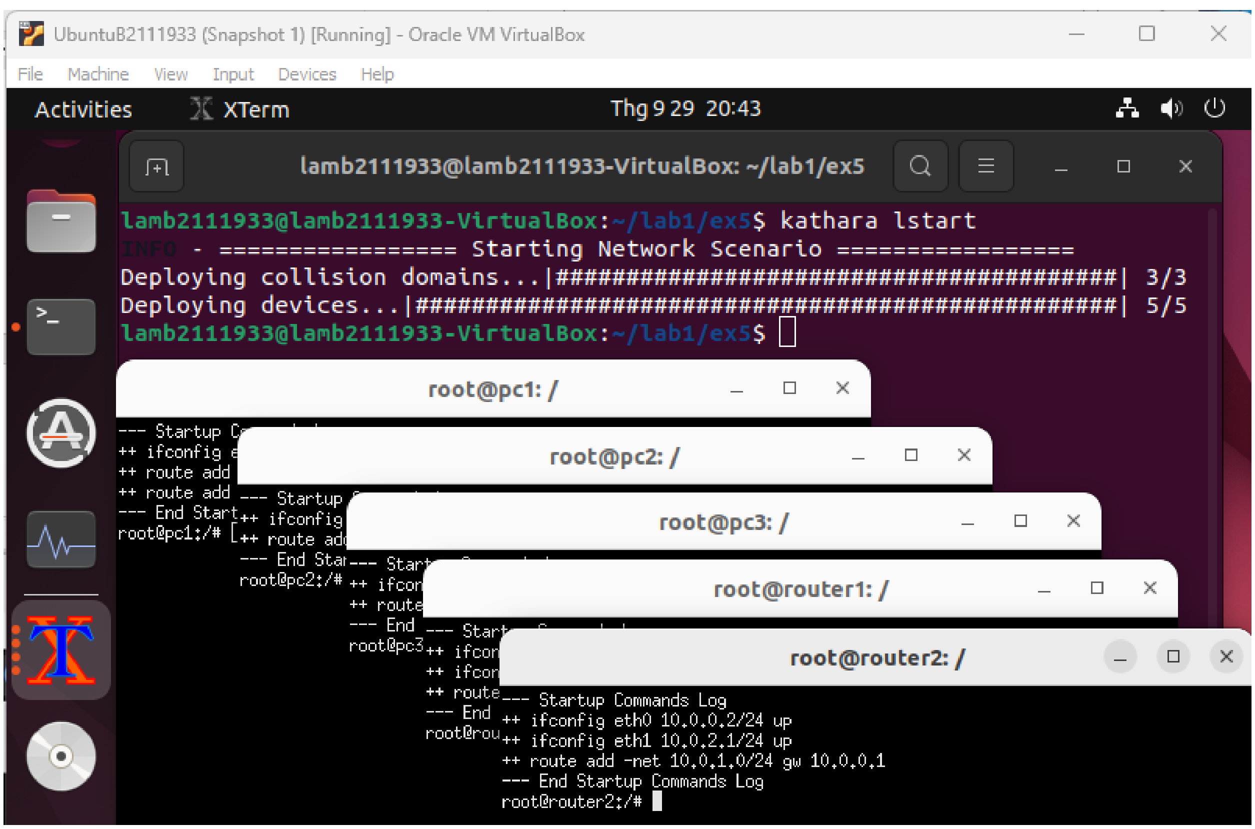


Construct the network:

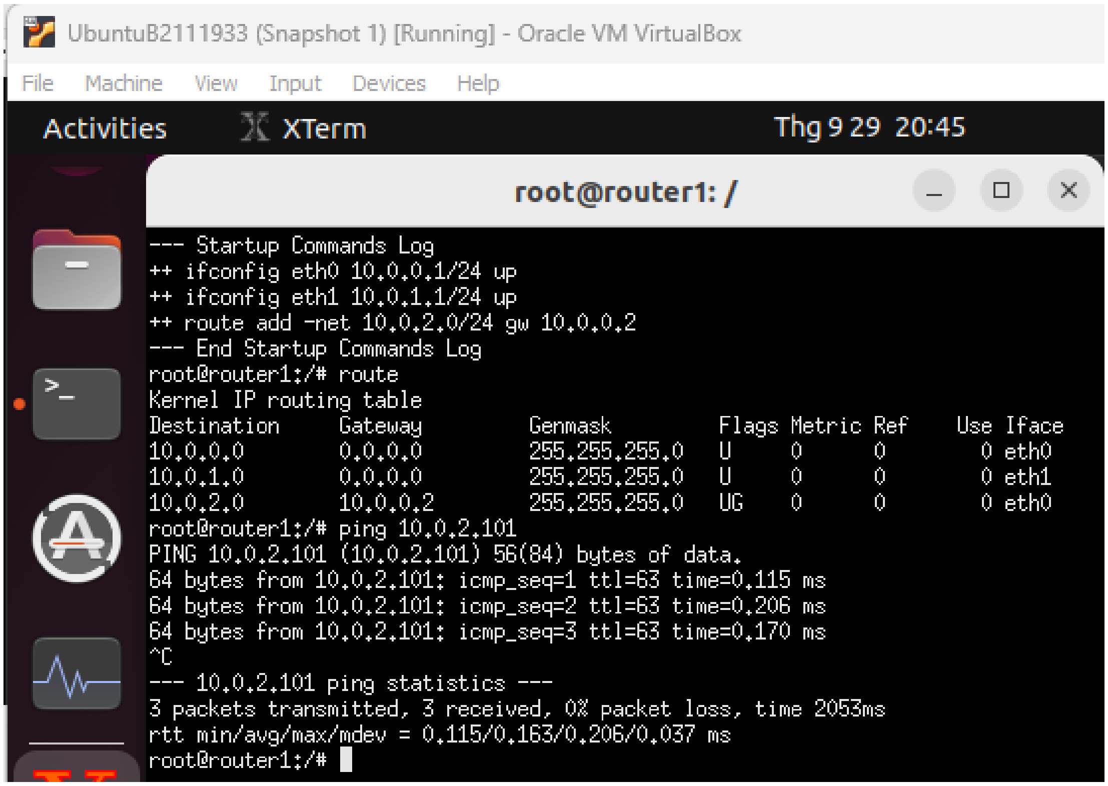


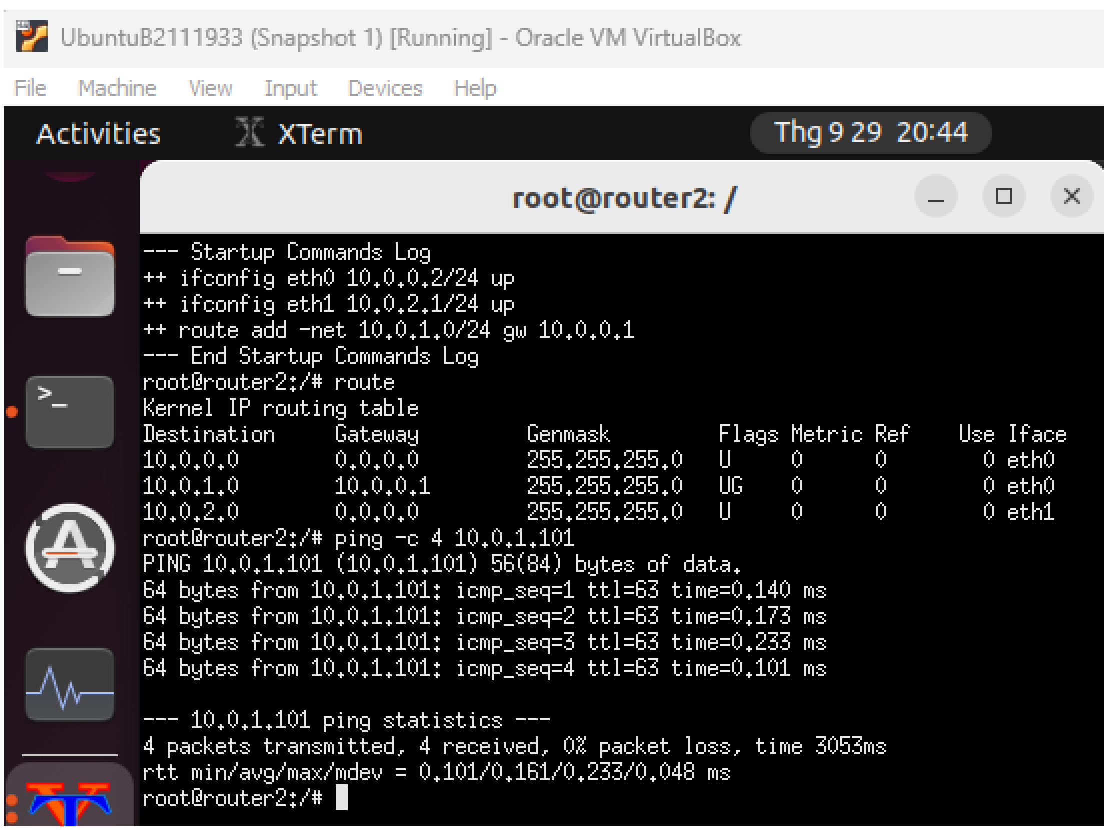


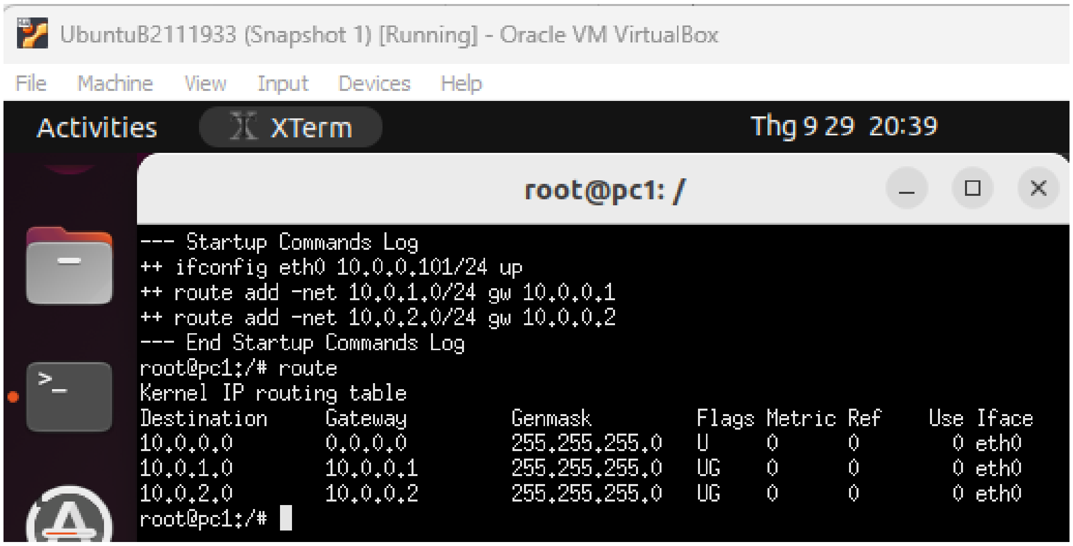
Start the lab:

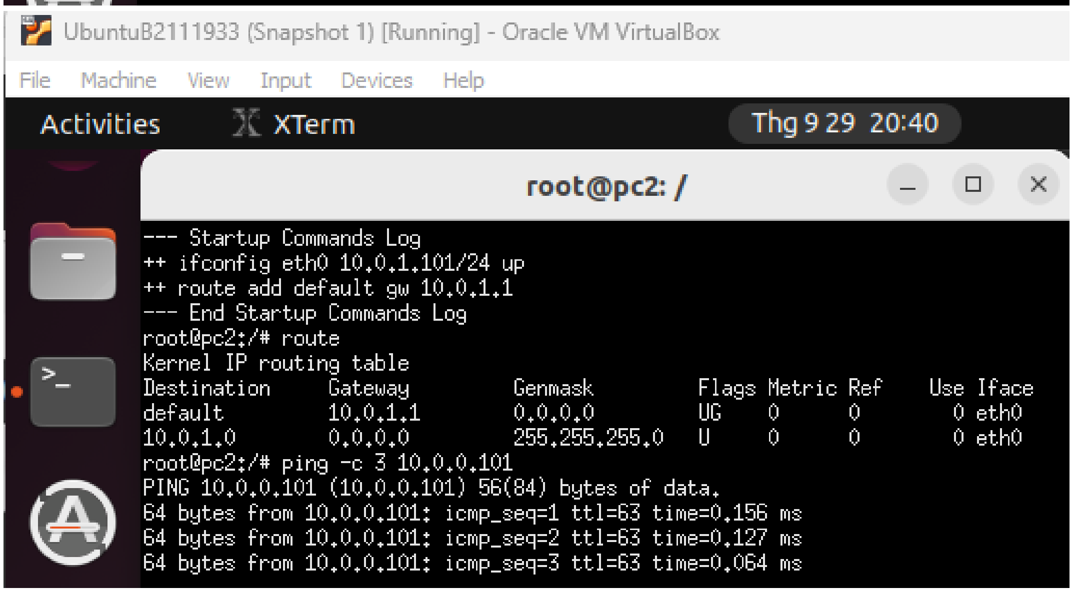


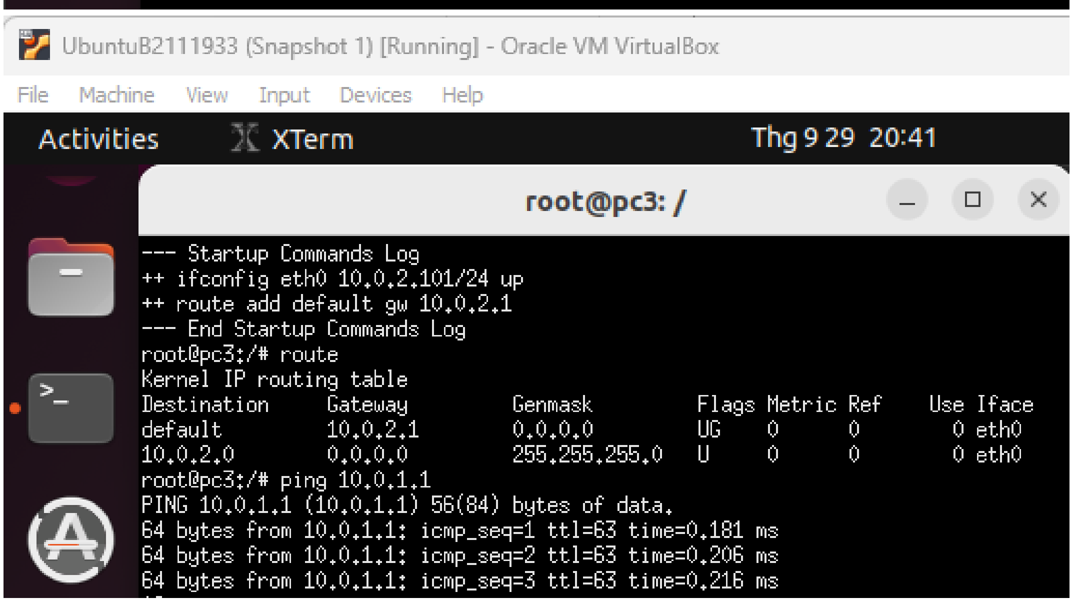
Check the result:





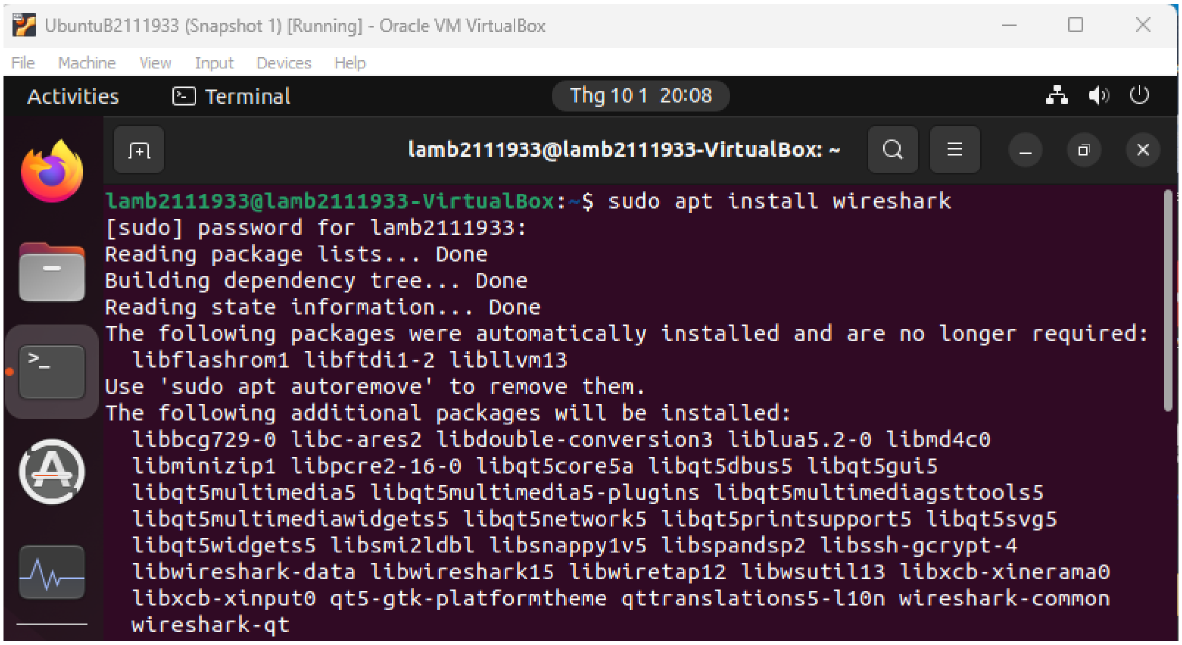

UbuntuB2111933 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities XTerm Thg 9 29 20:39
root@pc1: /
--- Startup Commands Log
++ ifconfig eth0 10.0.0.101/24 up
++ route add -net 10.0.1.0/24 gw 10.0.0.1
++ route add -net 10.0.2.0/24 gw 10.0.0.2
--- End Startup Commands Log
root@pc1:/# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
10.0.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
10.0.1.0 10.0.0.1 255.255.255.0 UG 0 0 0 eth0
10.0.2.0 10.0.0.2 255.255.255.0 UG 0 0 0 eth0
root@pc1:/#


UbuntuB2111933 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities XTerm Thg 9 29 20:40
root@pc2: /
--- Startup Commands Log
++ ifconfig eth0 10.0.1.101/24 up
++ route add default gw 10.0.1.1
--- End Startup Commands Log
root@pc2:/# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
default 10.0.1.1 0.0.0.0 UG 0 0 0 eth0
10.0.1.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@pc2:/# ping -c 3 10.0.0.101
PING 10.0.0.101 (10.0.0.101) 56(84) bytes of data.
64 bytes from 10.0.0.101: icmp_seq=1 ttl=63 time=0.156 ms
64 bytes from 10.0.0.101: icmp_seq=2 ttl=63 time=0.127 ms
64 bytes from 10.0.0.101: icmp_seq=3 ttl=63 time=0.064 ms

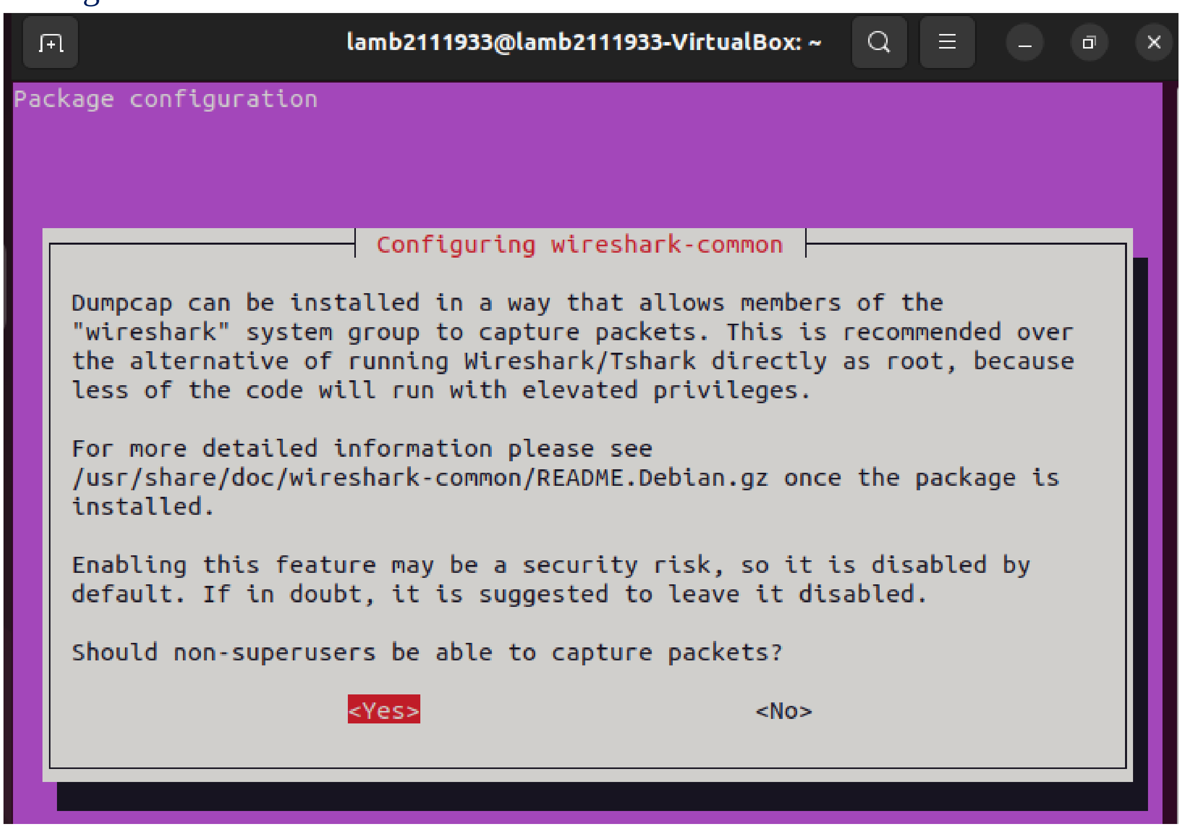

UbuntuB2111933 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities XTerm Thg 9 29 20:41
root@pc3: /
--- Startup Commands Log
++ ifconfig eth0 10.0.2.101/24 up
++ route add default gw 10.0.2.1
--- End Startup Commands Log
root@pc3:/# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
default 10.0.2.1 0.0.0.0 UG 0 0 0 eth0
10.0.2.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@pc3:/# ping 10.0.1.1
PING 10.0.1.1 (10.0.1.1) 56(84) bytes of data.
64 bytes from 10.0.1.1: icmp_seq=1 ttl=63 time=0.181 ms
64 bytes from 10.0.1.1: icmp_seq=2 ttl=63 time=0.206 ms
64 bytes from 10.0.1.1: icmp_seq=3 ttl=63 time=0.216 ms

Use Wireshark:

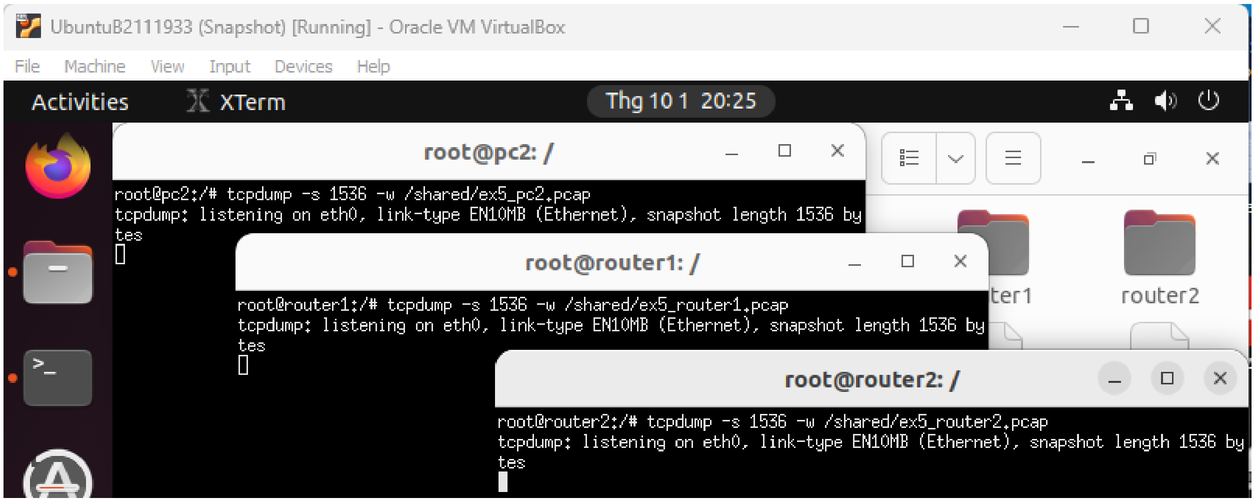
Install Wireshark



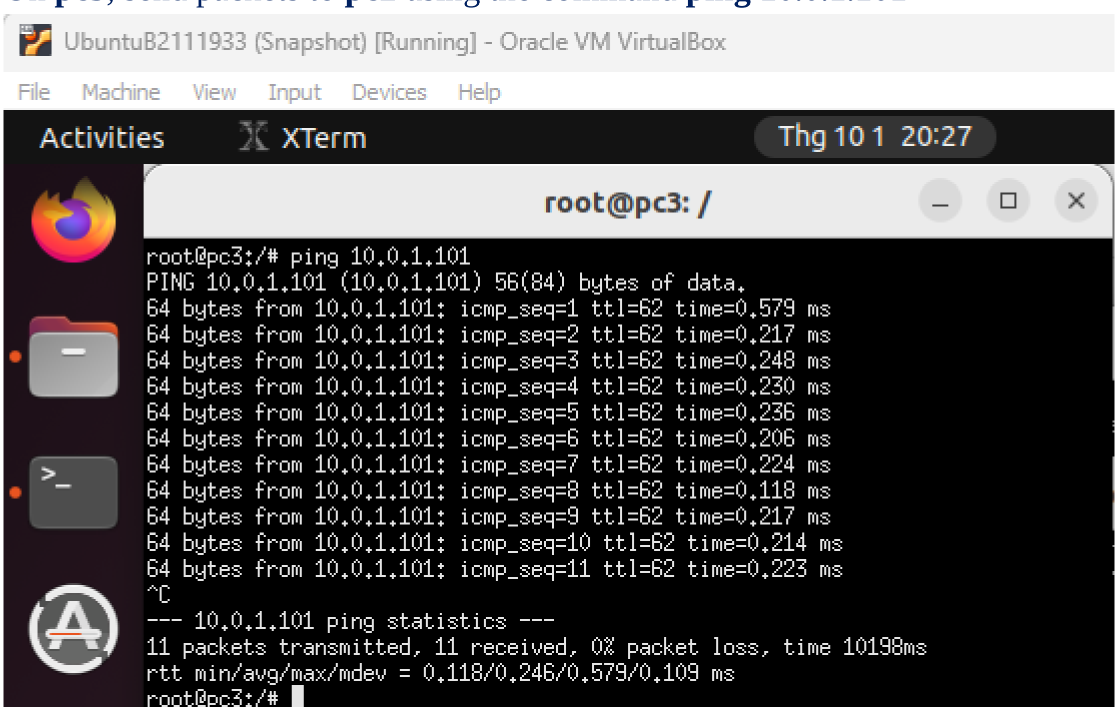
Configure Wireshark



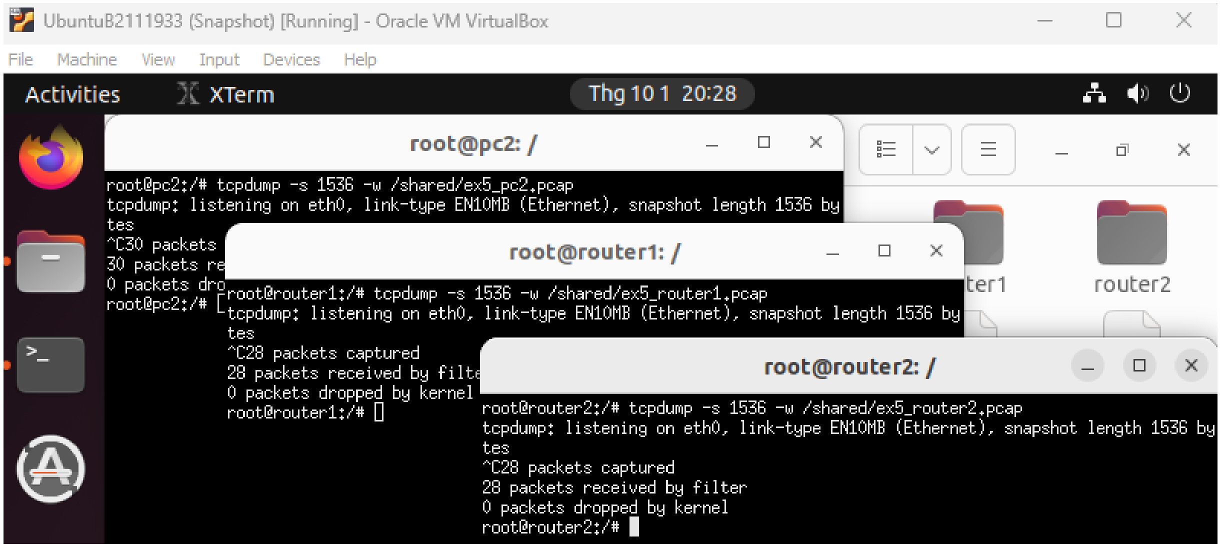
All packets are save in **.pcap** files which are in the **/shared** folder



On **pc3**, send packets to **pc2** using the command **ping 10.0.1.101**



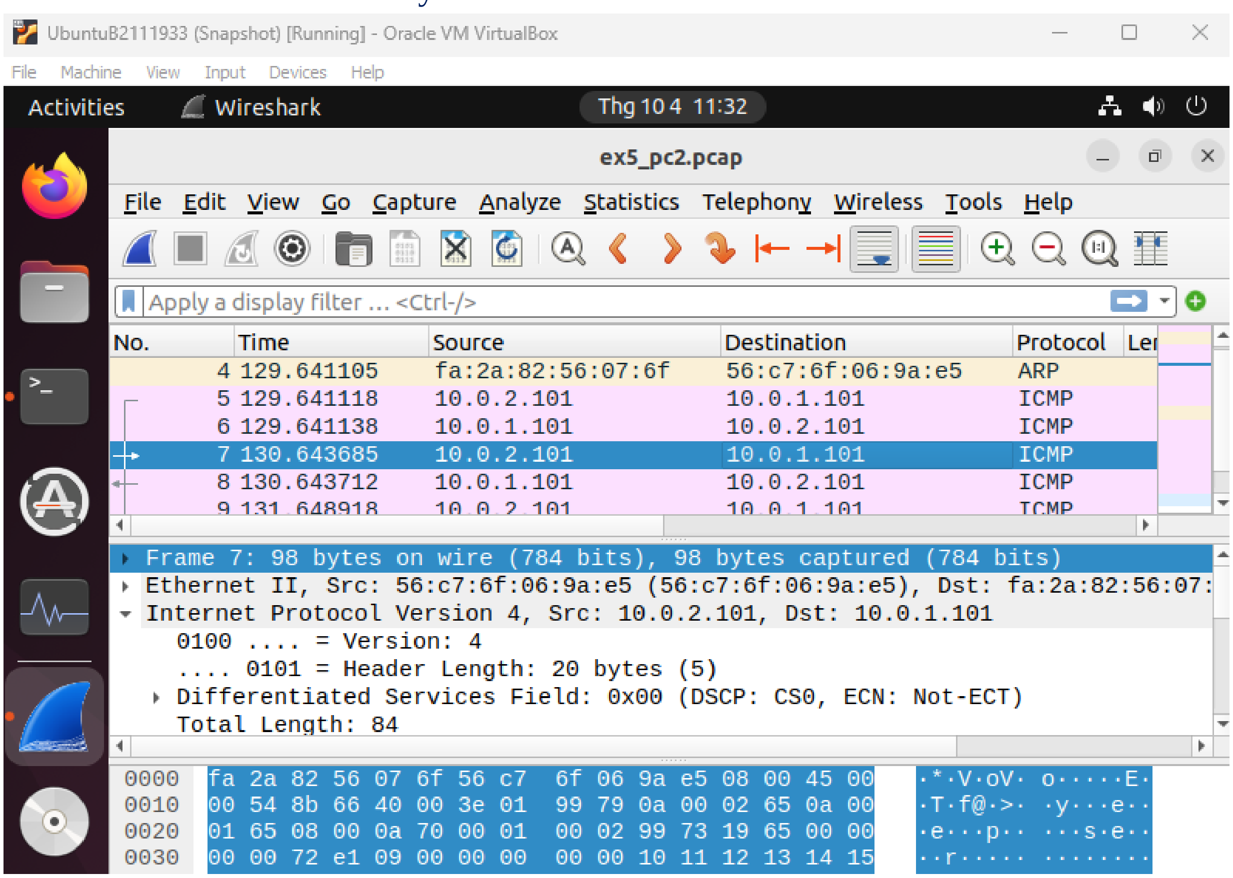
Stop the **tcpdump** on **pc2**, **router1** and **router2**



On the Ubuntu, open Ex5_pc2.pcap using Wireshark, select the frame #7 and answer the following questions:

- Size of frame in bytes?

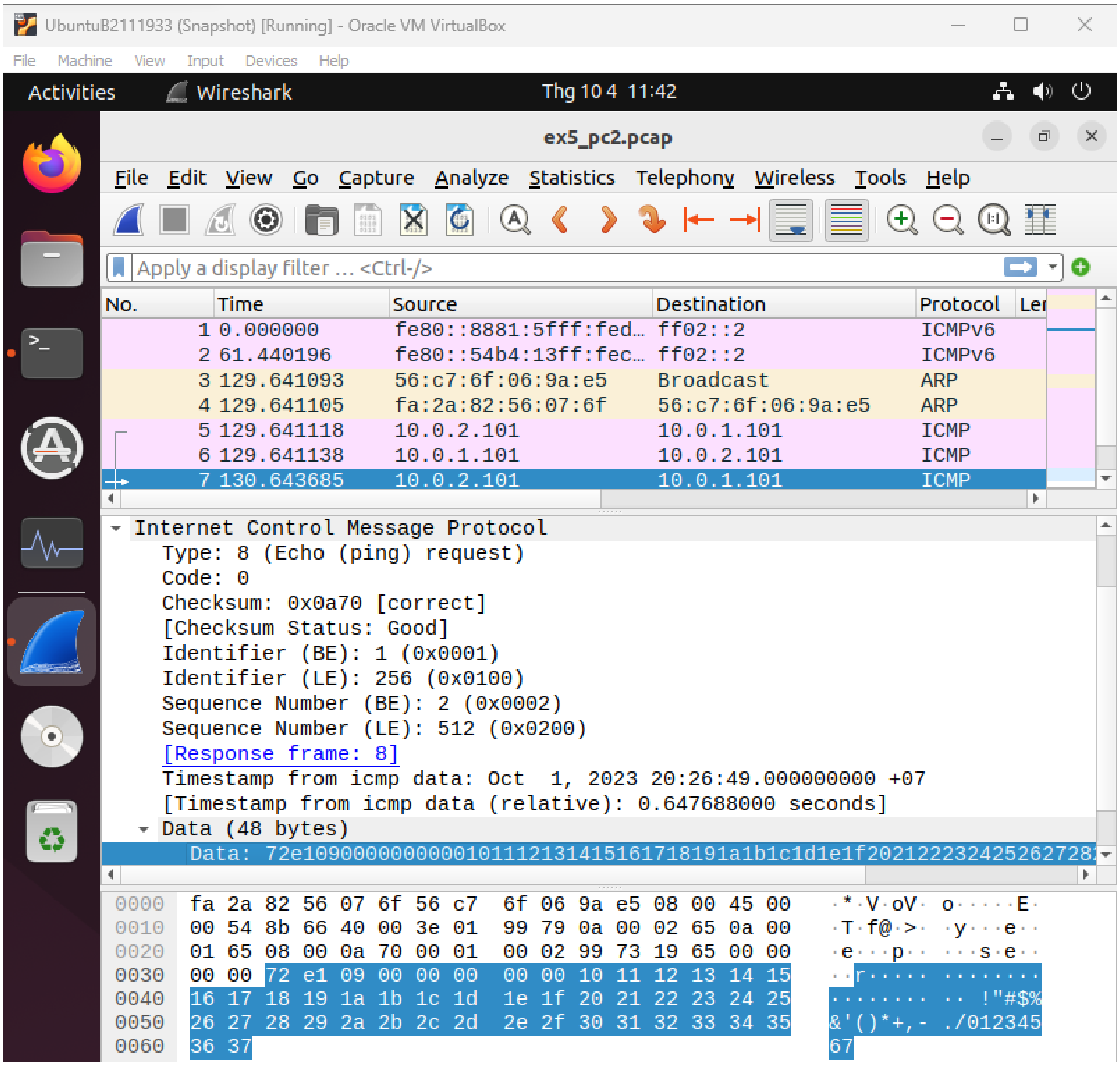
Answer: Size of frame is 98 bytes



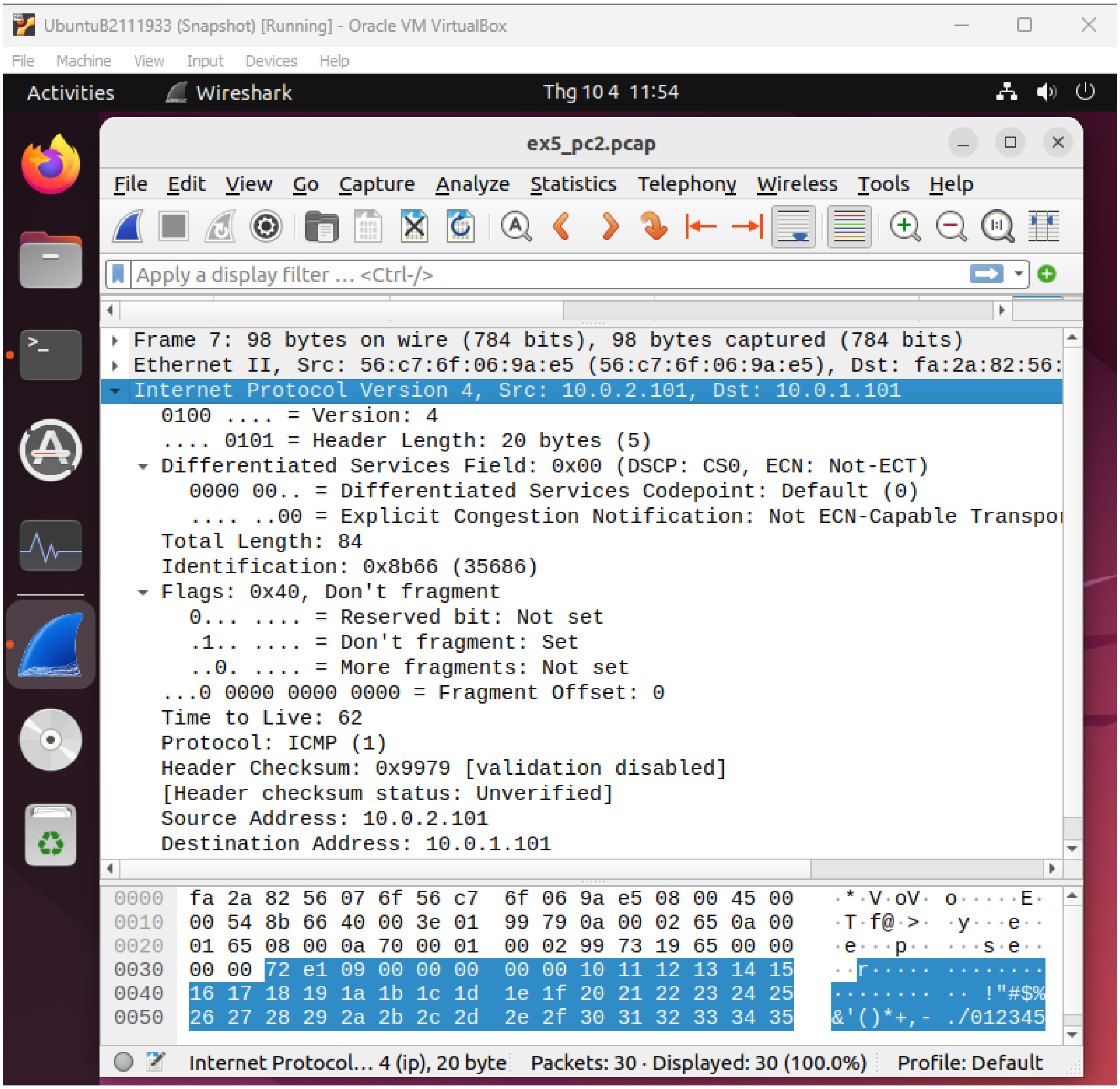
- Select Header Internet Control Message Protocol → which protocol is using? On which layer of the OSI model does this protocol operate? What is the content of the message? How long is this message in bytes?

Answer: - The protocol is **ICMP**

- This protocol operates on **network layer**
- The content of the message is below
- Size of this message is **48 bytes**



Select Header Internet Protocol Version 4:



→ what are the IP addresses of the source and destination hosts?

Answer: IP addresses: source: **10.0.2.101** destination: **10.0.1.101**

→ What is the length of the IP packet header? What fields does the Header include? How long is each field (Bytes)

Answer: - The length of the IP packet header: 20 bytes

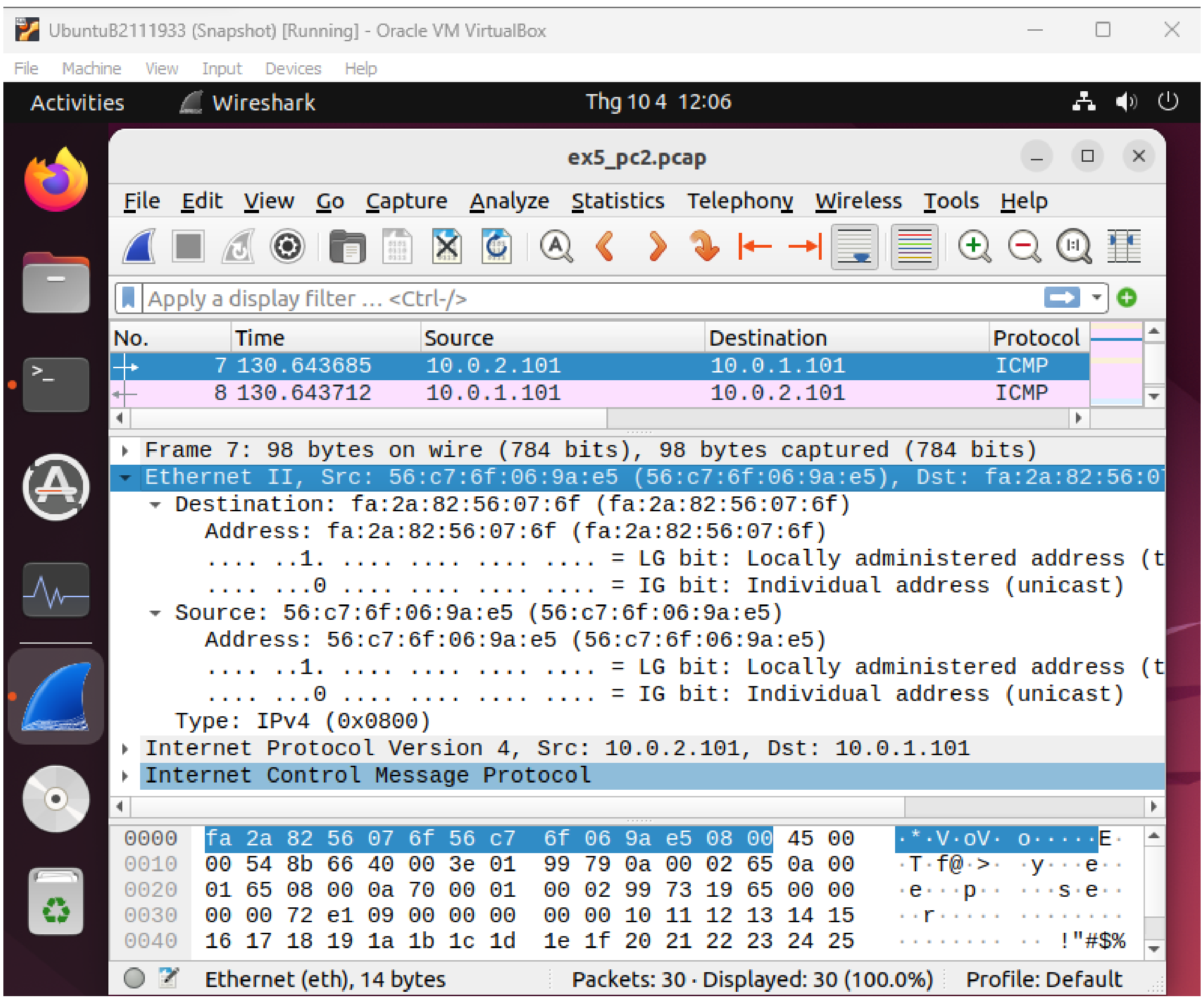
- Fields of the Header:
  - Version & Header Length (1 byte)
  - Differentiated Services Codepoint & Explicit Congestion Notification (1 byte)

- Total Length (2 bytes)
- Identification (2 bytes)
- Flags (2 bytes)
- Time To Live (1 byte)
- Protocol (1 byte)
- Header Checksum (2 bytes)
- Source Address (4 bytes)
- Destination Address (4 bytes)

→ What is the length of the Total Length field (Bytes).

Answer: the length of the Total Length field is **2 bytes**

Select Header Ethernet II:



→ What are the MAC addresses of the source and the destination hosts?

Answer: MAC addresses: - source: 56:c7:6f:06:9a:e5
- destination: fa:2a:82:56:07:6f

→ What is the Type value?

Answer: Type: IPv4 (0x0800)